Universidad
Carlos III de Madrid
www.uc3m.es

Extracción de información de las carreteras para la reducción del consumo de combustible y mejora en la seguridad

---

Trabajo Fin de Grado

2014-2015

Grado en Ingeniería en Tecnologías de Telecomunicación

Departamento de Telemática

Autor: *Vicente Gallardo Cabrera*

Tutor: *Víctor Corcoba Magaña*

Leganés, 9 de octubre de 2015

Agradecimientos

Con este proyecto se cierra una etapa de mi vida para dejar paso a abrir otras muchas más que, espero, sean aún mucho mejores.

Debo dedicar este proyecto a todas las personas que me han estado conmigo desde siempre, y también a aquellas que, por circunstancias de la vida, ya no están para poder dedicarles esto.

Una mención especial a mis padres, sin ninguno de los cuales hoy estaría aquí.

También dedicarles esto a todos aquellos que he ido conociendo a lo largo de la vida, ya sean amigos o compañeros, que me han hecho ser como soy hoy.

Mencionar también a mi tutor Víctor, que me ha ayudado en todo lo que ha estado en su mano en este proyecto y me ha animado a seguir con él.

A todos aquellos que han sido importantes para mí y yo para ellos.

A todas aquellas personas que aún no conozco y que, gracias a cómo soy hoy, conoceré.

Muchas gracias a todos.

Resumen

En la actualidad es de especial relevancia el tráfico urbano e interurbano, y el auge del pensamiento de las Smart Cities [1]. Con motivo de todo esto, se ha planteado crear una aplicación que pueda detectar y reconocer señales de tráfico *al vuelo* para avisar al conductor ayudándole a la conducción disminuyendo el número de accidentes, el consumo del vehículo, reduciendo las infracciones y el estrés y, como complemento, obtener información de las vías para su posterior procesamiento.

Gracias al auge de las tecnologías móviles y, cada vez, de su mayor capacidad de respuesta y almacenamiento, se ha decidido implementar esta aplicación en dispositivos móviles Android para que pueda ser utilizada por casi cualquier persona, con un escaso coste y un gran beneficio para la misma y para los de su alrededor.

A lo largo de esta memoria se expondrán todos los análisis previos, valoraciones, pasos a seguir e inconvenientes que han surgido para la realización de esta aplicación, con el fin de ayudar a comprender mejor a la misma y de contribuir a las distintas líneas de investigación que hay para éste y similares propósitos.

Extended Abstract

There is a growing problem with the increase in car traffic in cities and on roads in the short, medium and long term. Because of this, in order to improved road safety and reduce traffic problems is necessary to have more control over traffic signaling.

In order to solve the problem of urban and interurban traffic, programs among various public and private institutions are being designed, working on future Smart Cities [1] that use information and communications technology as the basis for the solution. A clear example of the importance of traffic signs detection and recognition is Google Self-Driving Car Project [2], which allows self-driving by applying various techniques, such as signal recognition and object detection.

As we can see, Traffic Sign Recognition (TSR) is really important and interesting for automotive market, industry and local authorities in charge of implementing signal maintenance. On this basis, an appropriate signal identification from vehicles passing on urban and interurban roads would provide driving assistance that increases safety and, as a significant extra, reduces fuel consumption. [3]

All this has promoted real motivation to provide driving support, reducing the number of accidents, decreasing fuel consumption, reducing traffic violations and also reducing driver stress and getting information from roads where vehicles travel. Thus, the main objective of the project is limited to extracting information from roads, based on detection and recognition of vertical road signs, in order to reduce fuel consumption and improve security.

In order to make all the above possible, we have decided to implement a mobile application capable of recognizing and displaying traffic signals and then storing a signal id and its geoposition in an effective, efficient, simple way with the ability of easy installation on any mobile based on Android OS.

Due to poor training on image processing as well as an initial poor concise final solution, we decided to follow a spiral development process to carry out this project, more suitable for this type of projects.

In "State of the art" section several applications has been analysed, for both Android devices (myDriveAssist [4] and DriveAid [5]) as embeded systems (Google Self-Driving Car Project [2], Volvo Road Sign Information (RSI) [6] and BMW Traffic Sign Recognition [7]).

After studying applications and systems mentioned in the previous paragraph we realized many of them did not work properly and/or they were not available or easy to obtain for general public.

The choice of Android is due to a previous study of the mobile technology sector, where we could see Android is the most used operating system worldwide for both smartphones and tablets.

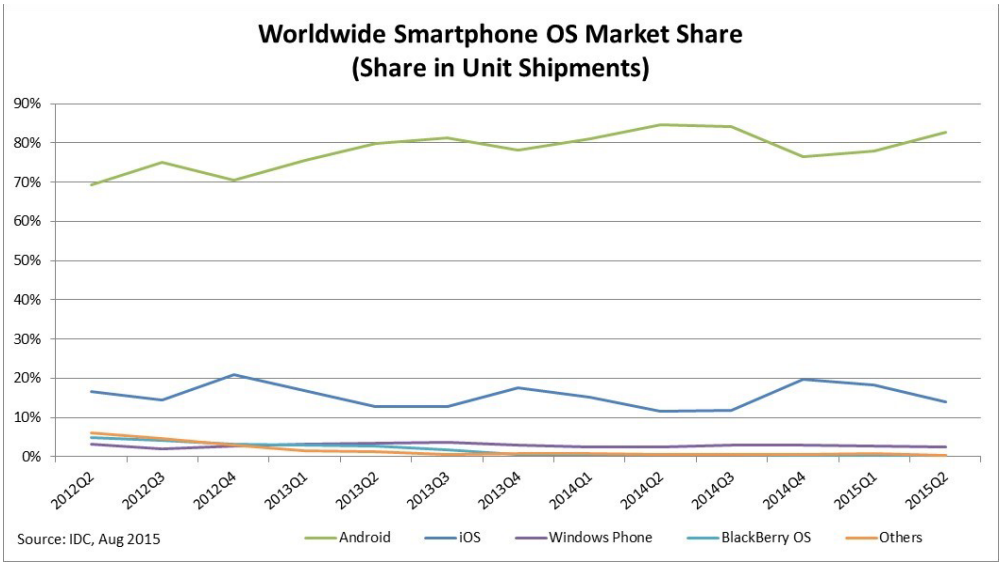


Illustration 1. Worldwide Smartphone OS Market Share [8]

The choice of mobile devices has not been random as they currently contain hardware that is not included as standard on devices such as laptops. The ability of current mobile devices to have implemented cameras and GPS systems in hardware and software were decisive; just as the Internet connection, common to desktop computers.

The application has been developed on Windows 7, where NVIDIA Tegra Android Development Pack (TADP) [9] has been installed. TADP is a development application pack created by Nvidia Corporation for Android, and it is intended for developers of videogames and applications with a high load of image processing on Android devices. TADP consists of several tools for developing on Android, such as SDK, NDK, JDK, APIs and Eclipse with a wide variety of plugins. It also includes OpenCV libraries, used for image processing.

OpenCV is an open source artificial vision computer library. It's written in C and C++ and it works under Linux, Windows and Mac OS X. There is also an active development for Python, Ruby, Matlab and other languages like Java. In this project, we will use the variation implemented in Java for Android platforms. OpenCV will be used for image processing and applying filters and descriptors. The images will be obtained through the mobile camera.

Keeping in mind the need for storing signaling identifying data and geopositional data we decided to use SQLite library for Android, so we can store data in a simple way using a relational database.

It is particularly important for image processing to define and explain the use of filters and algorithms applied to image processing.

The most important filter used is Gaussian blur filter, in which values are pixel colours and pixel position in its bidimensional image spatial location. These values are mixed with those positions next to the image, mixing colours and, so, producing image blurring. This behaviour would be similar to that in a myopic over long distances or a hyperopic over short distances.

One of the most significant algorithms would be the Canny Edge Detector algorithm. It is also known as the optimum detector. It satisfies the criteria of having a low error rate in detecting real edges, with a good location. It is based on obtaining a single point for each detected real edge.

For all this, it is necessary to reduce the noise with the Gaussian blur filter, find the intensity gradient in the image, delete items that are not in a maximum value and meet certain requirements of closeness to another point within a hysteresis threshold. In summary, this algorithm detects the edges of the figures. It would be similar to a high pass filter in image processing, which would pass the high frequencies of an image, corresponding to the edges.

Another important algorithm for detection of forms is the Hough Transform algorithm (Hough Transform) which is used to determine the most relevant edges for circular, square and triangular forms, essential part of this project. This transform operates by assuming that, for n

points in an image, you want to find subsets of these points on straight lines of the form $y = ax + b$ or, in the case of circles, lines following the circle equation $(x - c_1)^2 + (y - c_2)^2 = {c_3}^2$.

After obtaining these forms, we need to compare them to other forms preloaded for recognition. This is a task for the ORB descriptor, responsible for identifying the image keypoints contained within the shape detected by the Hough Transform algorithm (circle, triangle and square) and the application preloaded images. ORB then proceeds to use these keypoints to compare images and obtain sample matching pairs and their relative distances from the source image, which is one of the preloaded images. An example of how ORB works is in Illustration 2. Traffic signals compared with matching points find and linkedIllustration 2.

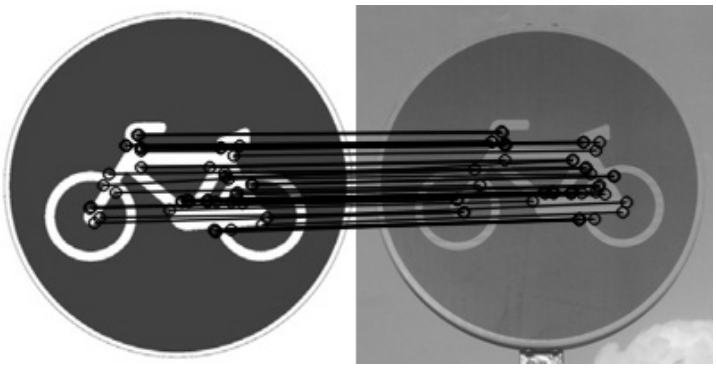


Illustration 2. Traffic signals compared with matching points find and linked

As an initial option Cascade Classifier algorithm, which uses the "Attentional" cascade, was chosen instead of ORB. This "attentional" cascade is a degenerated decision tree where, in each phase, a detector is trained to identify almost all objects of interest while rejecting those who are not. This algorithm was discarded due to the difficulties in processing the XML files generated and the various bugs that Android's OpenCV implementation has.

The final developed application is called DSTr (“Traffic Signal Detector” or "Detector de Señales de Tráfico" in Spanish) and allows the detection of circular, triangular and square signals using the above algorithms. Also, due to the need to store identification data and geopositional data (latitude and longitude), a relational database is created for storing the signals detected by images captured by the camera, provided the GPS and/or the Internet connection are enabled on the device.

The external appearance of the application is very simple, to get a friendly-user interface. It consists of a screen with the name of the application and then a drop down menu containing the functions divided into two parts: signal detection and management of the database. Bear in mind that the application works in landscape mode because of certain limitations of OpenCV.

Illustration 3. First selection screen of the application

The internal design of the application is more complex, since it has the feature to capture images using the camera device (smartphone or tablet) and process them in order to detect and extract images from recognizable figures and compare them with traffic signals preloaded and then obtaining the matches and storing the identified ones with respect to their identifiers and their geoposition data (latitude and longitude). We have also implemented the ability to clean and export the database. A class diagram has been added for a better explanation on how the application is made and several flow charts containing operation flows related to the detection and recognition of the traffic signals selected. Moreover, the design of the database application is also shown.

Tests have been basically form detection and sign recognition. These tests were performed on a Sony Xperia Z1 smartphone with image capture resolution of 720x480dpi. As for detection, tests performed better with circles, then triangles and squares respectively. In the case of recognition of signals, the best results have been obtained for signals with are more drawing complexity, since the descriptor ORB can get more keypoints and is able to obtain a better compare.

Based on all the above, the main goal to achieve was to develop an application capable of detecting vertical road signs in an effort to help the driver, either by improving driving,

reducing fuel consumption, reducing the number of accidents on the roads or releasing driver stress, and compatible with any device with Android operating system (smartphones or tablets). Obtaining a full simple and maintainable application which can be used on any Android device with a rear camera of an acceptable resolution has been one of the most important objectives throughout the project.

Regarding difficulties, the main obstacle to tackle was to understand current technologies applied (Android and OpenCV) and whether they would be appropriate or not for the project. Although at least as important was the amount of time used to learn that technologies and the difficulty to master them.

It is also noticeable the limitations of hardware, both PC and mobile device where the project was developed for most of the time (except for cascade detector training) lacked from great performance rates.

As a final conclusion, the assessment of the experience over the course of the project was very positive. The setbacks arisen have been also a good lesson to learn. It was a very comprehensive project, challenging and full of knowledge to acquired and retain. In addition, this line of research can be continued for future improvements.

Índice

Índice de ilustraciones

Índice de tablas

Índice de ecuaciones

1. INTRODUCCIÓN

1.1. Introducción

En este primer capítulo se va a dar una visión introductoria del proyecto, exponiendo el problema que se pretende resolver, la motivación y los objetivos del mismo, así como el hardware y el software utilizado para su realización y completitud. Además, se explicará la metodología usada para la resolución del problema y se dará un enfoque general de la estructura del contenido de ésta memoria.

1.2. Problema

Actualmente, el aumento del tráfico automovilístico en las ciudades y en las carreteras es un problema creciente a corto, medio y largo plazo. Debido a esto, es necesario tener un mayor control sobre la señalización para un mejor tránsito en las vías y disminuir los problemas de circulación y aumentar la seguridad vial. También es de destacar que, no sólo hay problemas en la circulación, sino también, sobre todo en ciudades, problemas de interacción con los viandantes, por lo que es necesaria una mayor atención a la señalización para evitar los problemas subyacentes de la interacción entre los propios vehículos, y entre éstos y las personas.

Con el fin de solucionar el problema del tráfico urbano e interurbano se están diseñando programas entre diversas instituciones públicas y privadas, trabajando en las futuras Smart Cities [1] que utilizan las tecnologías de la información y la comunicación como fundamento para esta solución. Un claro ejemplo de la importancia en la detección de señales de tráfico y reconocimiento de las mismas es el prototipo de coche autónomo de Google [2] , que permite conducir de manera autónoma aplicando diversas técnicas, tanto de reconocimiento de señales como de vía urbana.

Como se puede ver, el Reconocimiento de Señales de Tráfico (RST) es de gran relevancia e interés para el mercado automovilístico, el sector industrial y para las autoridades locales encargadas del mantenimiento de implantación de las señales. En base a esto, una correcta identificación de las señales desde los vehículos que transitan vías urbanas e interurbanas permitiría suministrar ayuda a la conducción aumentando la seguridad y, como añadido relevante, reduciendo el consumo. [3]

Por todas las causas expuestas anteriormente, se ha decidido implementar una aplicación móvil que solucione los problemas de RST de una manera eficaz, eficiente, sencilla y de fácil instalación en cualquier dispositivo móvil cuyo sistema operativo sea Android. Esta portabilidad y usabilidad facilitarán su uso entre los conductores, pudiendo ser así una ayuda

complementaria a sus capacidades de percepción, pudiendo ésta implantarse en cualquier tipo de vehículo (antiguo o moderno) que circule por cualquier tipo de vía.

1.3. Motivación del proyecto

Los motivos principales por los que se ha llevado a cabo éste proyecto son:

- **Ayudar a la conducción**, mostrando la señalización de la vía al conductor en un dispositivo móvil con el fin de que realice una conducción más segura y eficiente.
- **Disminuir el número de accidentes** provocados por el despiste de conductores al no observar algunas señales en el transcurso de su conducción.
- **Reducir el consumo del vehículo,** debido al aviso previo de la señalización, lo que motiva una conducción menos brusca y, por lo tanto, un menor uso de combustible.
- **Reducir las infracciones** producidas por los conductores desde sus vehículos.
- **Reducir el estrés del conductor** gracias a que puede ser informado previamente de la señalización siendo capaz de regular suavemente la velocidad, evitando así frenazos y teniendo una conducción más tranquila.
- **Obtener información de las vías** para tener un plano global de su señalización y mejorar la misma.

Asimismo, hay que destacar que la elaboración de este proyecto también es llevada a cabo como opción complementaria para disminuir el número de infracciones y accidentes a las soluciones que ya está llevando a cabo la Dirección General de Tráfico y campañas como PONLE FRENO del grupo Atresmedia que, siempre, a pesar de todos los esfuerzos, parecen ser insuficientes.

1.4. Objetivos

El principal objetivo de éste proyecto es desarrollar una aplicación que extraiga información de la carretera, concretamente, que **detecte y reconozca señales de tráfico verticales**. Para ello se decide desarrollar la aplicación en plataforma Android para que sea accesible desde cualquier dispositivo móvil (smartphone o tableta) compatible con dicho sistema operativo.

La aplicación deberá detectar y reconocer señales de tráfico *al vuelo* con el fin de mostrarlas al conductor y complementar así la capacidad visual de éste con la información que le pueda aportar el dispositivo móvil. También deberá almacenar las señales de tráfico

reconocidas, con el fin de identificar los puntos de la vía en los que se encuentran y complementar así la información de geoposición (latitud y longitud) en las que se encuentren éstas.

Cumpliendo con lo anterior, se realizará una interfaz para el usuario para que pueda elegir el modo detección y reconocimiento, mostrando así las señales que prefiera.

Así pues, se pretende incrementar la seguridad en las vías de circulación y reducir el número de infracciones de tráfico mediante la implantación de ésta aplicación, así como obtener información sobre las señales de tráfico verticales ya existentes en las vías urbanas e interurbanas.

1.5. Metodología de resolución del problema

Para la realización de éste TFG, debido a la escasa formación sobre procesamiento de imagen tanto como a una solución final del proyecto poco concisa inicialmente, se ha decidido seguir un proceso de desarrollo en espiral. Esto se decidió después de evaluar otras metodologías como el ciclo de vida tradicional, en cascada y la metodología SCRUM.

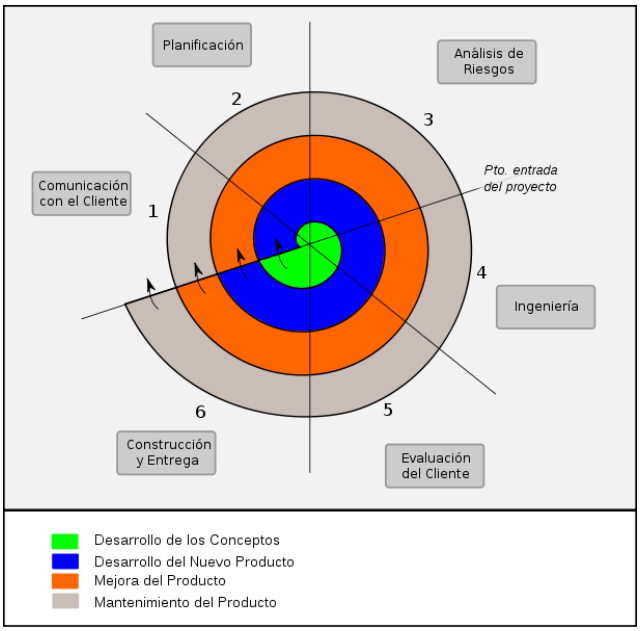


Ilustración 4. Ciclo de vida en espiral de Boehm [10]

La metodología de desarrollo en espiral consiste en ir añadiendo funcionalidades al sistema, siguiendo las indicaciones del cliente o usuario final que, en este caso, es el propio desarrollador. De este modo se van completando y resolviendo las especificaciones dadas

inicialmente y se van ajustando las funcionalidades a los requisitos y necesidades que surgen, sobre todo, de la interacción con el software, aunque no menos de la interacción con el hardware.

Debido pues, a la capacidad que tiene este tipo de metodología de realizar iteraciones, ha favorecido el seguir diferentes líneas de pruebas para llegar a una mejor solución, realizando así modificaciones con iteraciones y mejoras con respecto a éstas en nuevas iteraciones, llegando a versiones más consistentes y elaboradas del producto, con mejoras sustanciales respecto a versiones anteriores.

1.6. Contenido de la memoria

En esta sección se incluirá una descripción general de la estructura del presente documento, detallando cada uno de los capítulos que se definirá a lo largo del mismo.

En el primer capítulo se va a dar una visión introductoria del proyecto, exponiendo el problema que se presente resolver, la motivación y los objetivos del mismo, así como el hardware y el software utilizado para su realización y completitud.

Posteriormente, en el segundo capítulo se procederá a exponer el estado del arte relativo al proyecto planteado.

A continuación, en el capítulo tercero se expondrá el sistema operativo para el que se ha desarrollado la aplicación, haciendo énfasis en el entorno de desarrollo de éste.

En el cuarto capítulo se expondrán conceptos básicos de imagen digital, para mejor comprensión de este proyecto.

Seguidamente, en el quinto capítulo se expondrán algunos de los filtros, algoritmos y descriptores para imágenes digitales usados para la resolución del proyecto.

En el capítulo sexto se realizará un análisis de la aplicación, describiéndola, exponiendo sus casos de uso y sus requisitos de sistema.

Posteriormente, en el capítulo séptimo se expondrá el diseño de la aplicación, tanto externo como interno, mostrando su diagrama de clases, diagramas de flujo y el diseño de la base de datos.

En el capítulo octavo se exponen las pruebas realizadas de la aplicación con el dispositivo móvil de pruebas, exponiendo los resultados obtenidos.

Seguidamente, en el capítulo noveno se expondrán las conclusiones a las que se ha llegado y las líneas futuras de desarrollo del proyecto.

Tras ello, se añadirán todas las referencias consultadas durante la elaboración de este trabajo, que amplían la información citada a lo largo de todo el documento.

Finalmente, se ofrecen distintos anexos que ayudarán a la comprensión y completitud de la información de este documento. El primer anexo corresponde a los acrónimos y definiciones utilizados en toda la memoria. El segundo anexo corresponde a la gestión del proyecto. El tercer anexo indica la codificación utilizada para las señales de tráfico dentro del proyecto. El cuarto anexo nos muestra las especificaciones del dispositivo Android utilizado. Por último, el quinto anexo equivale al manual de usuario que explica el funcionamiento del sistema.

1. INTRODUCTION

1.1. Introduction

This first chapter pretends to provide an introductory overview of the project, exposing the problem, motivation and objectives, as well as hardware and software used for its achievement. In addition, the methodology used to solve the problem and a general approach to the structure of the content of this project memory will be explained.

1.2. Problem

Nowadays, there is a growing problem with the increase in car traffic in cities and on roads in the short, medium and long term. Because of this, in order to improved road safety and reduce traffic problems is necessary to have more control over traffic signaling. Nonetheless, problems are not related only to circulation, but also, especially in cities, interaction with pedestrians. Therefore, to prevent the underlying problems of interaction between the vehicles themselves, and between them and people is necessary greater attention to the signaling.

In order to solve the problem of urban and interurban traffic, programs among various public and private institutions are being designed, working on future Smart Cities [1] that use information and communications technology as the basis for the solution. A clear example of the importance of traffic signs detection and recognition is Google Self-Driving Car Project [2], which allows self-driving by applying various techniques, such as signal recognition and object detection.

As we can see, Traffic Sign Recognition (TSR) is really important and interesting for automotive market, industry and local authorities in charge of implementing signal maintenance. On this basis, appropriate signal identification from vehicles passing on urban and interurban roads would provide driving assistance that increases safety and, as a significant extra, reduces fuel consumption. [3]

Because of all reasons above, we have decided to implement a mobile application that solves the problems of TSR in an effective, efficient, simple way with the ability of easy installation on any mobile based on Android OS. This portability and usability will facilitate its use among drivers, and may be an additional aid to their perception, making possible its implementation in any type of vehicle (old or new) that circulates on roads.

1.3. Project Motivation

The main reasons to carry out this project are:

- **Driving support,** showing the pathway signaling on a mobile device in order to perform a safer and more efficient driving.
- **Reducing the number of accidents** caused by the confusion of drivers because of not seeing some signs while driving**.**
- **Decreasing fuel consumption** of the vehicle due to the previous notice of signaling, which means a less abrupt driving and less fuel usage.
- **Reducing the infractions** caused by drivers.
- **Reducing driver stress** by knowing signaling in advance and therefore being able to regulate the speed, avoiding sudden breakings and having a quieter ride.
- **Getting information from roads** to have a global vision of signaling and how to improve it.

The development of this project is also a complementary option to other campaigns focused on reducing the number of traffic infractions and accidents, such as those accomplished by Traffic Department (DGT) or "PONLE FRENO" by Atresmedia group, because despite all efforts, they appear to be insufficient.

1.4. Objetives

The main objective of this project is to develop an application that is able to extract information from the road, specifically, to detect and recognize vertical road signs. To that end, it was decided to develop the application on Android platform accessible to any mobile device (smartphone or tablet) compatible with that operating system.

The application must detect, recognize and show traffic signals on the fly in order to complement the visual ability of the driver. It can also identify the points of the track where recognized traffic signals are and store the geo information (latitude and longitude) in which they are located.

Thus, the implementation of this application intends to increase safety on roads and reduce the number of traffic infringements as well as gathering information from existing vertical traffic signals on urban and intercity roads.

1.5. Problem solving methodology

After evaluating several software engineering methodologies such as cascade life cycle or SCRUM methodology we decided to follow a spiral development process to carry out this project, due to poor training on image processing as well as an initial concise final solution.

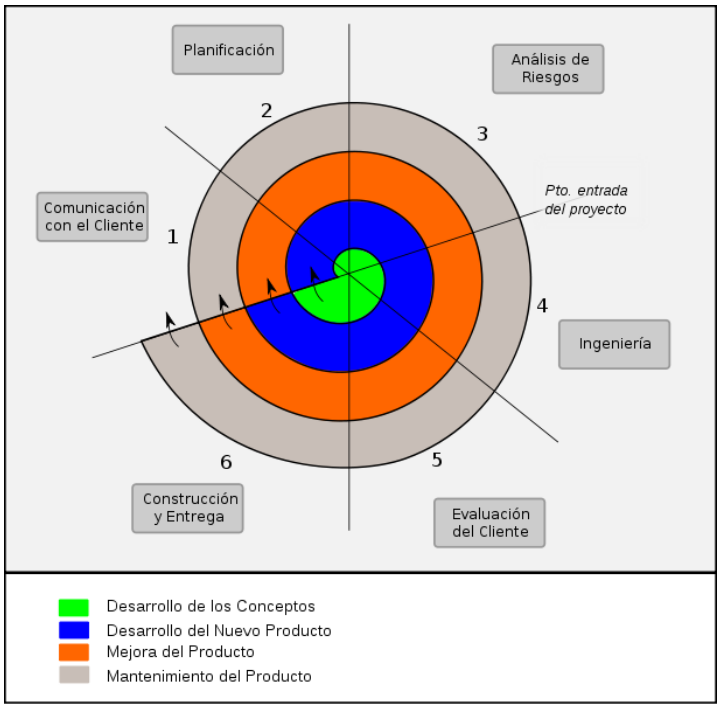


Ilustración 5. Boehm's spiral lifecycle model [10]

Spiral development methodology consists of gradually adding functionality to the system, following the instructions of the customer or end user (in this case the developer itself). Thus, initial given specifications are completed and resolved and functionalities are adjusted to the requirements and needs that arise with mainly software and hardware interaction.

The ability of this kind of methodology to perform iterations has allowed getting a better solution. And so, the changes and improvements between iterations have made possible to reach more consistent versions and a more elaborate product, with substantial improvements over previous versions.

1.6. Memory contents

This section provides an overview of the structure of this document, detailing the content from each chapter.

The first chapter pretends to provide an introductory overview of the project, exposing the problem, motivation and objective, as well as hardware and software used for its achievement.

Later, in the second chapter will proceed to expose the state of the art concerning the proposed project.

Then, in the third chapter will be exposed the operating system for which the application has been developed, focusing on the development environment.

In the fourth chapter will be exposed basics of digital image, for better understanding of this project.

Then, in the fifth chapter will be presented some of the filters, algorithms and digital imaging descriptors used for the resolution of the project.

In the sixth chapter will be analyzed the future application, describing, exposing their use cases and system requirements.

Later, in the seventh chapter will be exposed the application design, external and internal, showing his class diagram, flowcharts and database design.

In the eighth chapter will be shown tests of the application tested in the mobile, and the results of these ones.

Then, in the ninth chapter will be told the conclusions that have been reached and future lines of development of the project.

After that, will be added all the references consulted during the preparation of this work that will expand the information referred to throughout the document.

Finally, various annexes will be added to help understanding and completeness the information in this document. The first document corresponds to acronyms and definitions used throughout the specification. The second annex is for project management. The third annex indicates the encoding used for traffic signals within the project. The fourth annex shows the specifications from Android device used. Finally, the fifth annex is an equivalent to user manual that explains how the system works.

2. ESTADO DEL ARTE

2.1. Introducción

En este capítulo se procederá a exponer el estado del arte relativo al proyecto planteado. Se estudiarán las aplicaciones y sistemas similares a lo que se pretende conseguir, así como un diagnóstico de la situación actual, el entorno de desarrollo seleccionado y las conclusiones acerca del análisis.

2.2. Análisis de aplicaciones y/o sistemas similares

2.2.1. myDriveAssist

myDriveAssist [4] es una aplicación gratuita desarrollada para Android e iOS por BOSCH que, según su descripción de producto, detecta límites de velocidad, prohibición de adelantar y señales de fin de prohibiciones a través de la cámara del dispositivo móvil. También tiene la capacidad de enviar avisos cuando se rebase el límite de velocidad permitido o se esté en una zona de prohibido adelantar y, además, de enviar automáticamente los límites de velocidad detectados a la nube de myDriveAssist.

Ilustración 6. Logo de myDriveAssist [11]

Esta aplicación, tiene una visión agradable y una navegación atractiva e intuitiva.

Ilustración 7. Imagen de funcionamiento de myDriveAssist [11]

A pesar de todo lo anterior, en las pruebas realizadas con la misma no se ha podido detectar ninguna señal con la cámara del dispositivo utilizado. Parece ser que la aplicación no funciona de manera correcta con dispositivos Android superiores a la versión 4.2.

2.2.2. DriveAid

DriveAid [5] es una aplicación para dispositivos Android desarrollada por Enrique Mozos y Ramón Hervás en la Escuela Superior de Informática de Castilla-La Mancha. Según su descripción, permite detectar y reconocer señales de tráfico.

Ilustración 8. Captura de imagen de video de muestra de funcionamiento de DriveAid [12]

La aplicación no está disponible para el público en general, por lo que no se puede corroborar su funcionamiento ni explorar sus funcionalidades. No obstante, el propósito de detección es uno de los que persigue este mismo proyecto.

2.2.3. Google Self-Driving Car Project

El proyecto de coche autónomo de Google o Google Self-Driving Car Project [2] es un proyecto ambicioso llevado a cabo por la empresa Google con el fin de realizar un vehículo que se autoconduzca, detectando y reconociendo la señalización, tanto vertical como horizontal, personas, objetos, distancias y todo tipo de obstáculos e inconvenientes a los que se tiene que enfrentar cualquier conductor. Dispone de dispositivos láser, cámaras y otros tantos elementos para los propósitos anteriormente descritos.

Ilustración 9. Prototipo de coche autónomo de Google [2]

Sin embargo, a día de hoy, a pesar de todos los años que Google lleva dedicados a este proyecto, aún sigue siendo tal y no está a la venta, por lo que no se puede comprobar su eficiencia y eficacia.

2.2.4. Volvo Road Sign Information

Volvo ha desarrollado un sistema RST el cual denominan Volvo Road Sign Information (RSI) [6] , el cual consta de una cámara mirando hacia el frente, que explora el camino a seguir para detectar las señales de tráfico existentes. Esta cámara está conectada a un software de reconocimiento que, posteriormente, anota los cambios descritos en la señalización y transmite éstos al panel de instrumentos del coche.

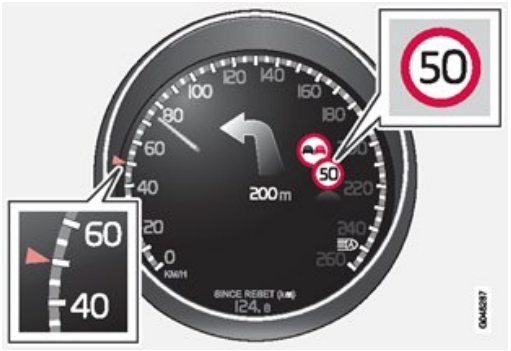


Ilustración 10. Funcionamiento en panel de instrumentos de Volvo RSI [6]

La información permanece en el panel de instrumentos hasta que se produce cualquier cambio, por lo que si un conductor no está seguro del límite de velocidad actual o cualquier otra señal puede revisarlo de una manera sencilla.

Este tipo de sistema está implantado en modelos como el Volvo S60 y el V40, no siendo apto para aquellos que no dispongan de estos vehículos.

2.2.5. BMW Traffic Sign Recognition

BMW [7] al igual que Volvo ha tomado la iniciativa de poner RST en sus vehículos de alta gama. Para ello ha introducido una cámara frontal en el retrovisor interior para registrar las señales de tráfico. En combinación con el sistema de navegación de serie incluye en la propia pantalla los límites de velocidad y la prohibición de adelantamiento.

La información registrada por la cámara se combina con datos de otros dispositivos como el sensor del sistema de navegación, el reloj y la lluvia antes de que se muestren las restricciones en el panel de instrumentos.

Ilustración 11. RST en automóvil BMW [13]

El sistema almacena todos los datos de forma temporal para que cuando reanude el viaje después de un breve descanso, los límites de velocidad y las señales de prohibición de adelantamiento válidas últimas en almacenarse se muestren inmediatamente al comienzo del viaje.

Este tipo de sistema lo podemos encontrar en automóviles BMW de las series 1, 5 y 7; por lo que no está disponible al público a no ser que se disponga de vehículos de alguna de estas series.

2.3. Diagnóstico de la situación actual

En la sección 2.1 se han analizado brevemente unas aplicaciones y sistemas similares relacionados con la identificación de señalización vertical de tráfico, y que conforman el estado del arte. Siguiendo ésta línea se ofrece una valoración de los requisitos más relevantes relacionados con la aplicación a desarrollar y que pueden ser de especial interés para la misma.

Las características más relevantes a destacar y a valorar son las siguientes:

- **Interfaz**. Se evalúa el atractivo de la interfaz y su funcionalidad, teniendo en cuenta su usabilidad y la capacidad intuitiva que puede suscitar en el usuario.
- **Detección y reconocimiento de señales verticales.** Se evalúa la capacidad de detección de la señalización y su correcto reconocimiento (capacidad de aciertos, errores, falsos positivos), así como su rapidez de respuesta.
- **Almacenamiento de datos.** Se evalúa si la aplicación o sistema realiza un registro de las señales detectadas para ampliar sus bases de datos y así realizar una mejora en la conducción implementando nuevas señales no registradas y actualizando las ya existentes.

La valoración de las características anteriormente expuestas se realizará por aplicación o sistema y se evaluará mediante el sistema decimal con números enteros comprendidos entre el 0 (menor puntuación) y el 10 (mayor puntuación). Es necesario indicar que algunas de las aplicaciones o sistemas no han podido ser probados debido a que no se ha dispuesto de ellos para la realización de las pruebas, por lo que se evaluarán como N/A (No Aplica) ya que sólo se puede hacer una evaluación a partir de los datos de informes o capturas de pantalla, y no de una evaluación práctica de los mismos.

Es de destacar que los análisis de las aplicaciones pueden, en cierto grado, ser subjetivos. No obstante, se ha procedido con la mayor objetividad posible a la hora de su realización para tener una visión lo más precisa posible del conjunto de las mismas.

La valoración de las aplicaciones y sistemas mencionados se puede encontrar en la siguiente tabla:

Aplicación y/o sistema	myDriveAssist	DriveAid	Google Car	Volvo RSI	BMW TSR
Interfaz	9	N/A	N/A	7	N/A
Detección y reconocimiento de señales verticales	0	N/A	N/A	5	N/A
Almacenamiento de datos	0	N/A	N/A	0	N/A

Tabla 1. Valoración de aplicaciones y/o sistemas similares

2.4. Entorno de desarrollo seleccionado

Debido al creciente auge de las tecnologías móviles en la actualidad y a su rápida evolución, tanto en portabilidad, escalabilidad y velocidad de almacenamiento y procesamiento con respecto a las tecnologías fijas como los ordenadores convencionales, de sobremesa y/o portátiles, se ha decidido realizar este proyecto sobre estas tecnologías móviles debido a estas características.

Gracias a lo expuesto en el anterior párrafo, se puede crear una aplicación portable, de fácil utilización, con una rapidez de respuesta alta y con una precisión lo suficientemente buena como para poder detectar formas y reconocer señales de tráfico.

Como complemento a lo anterior, estas nuevas tecnologías móviles nos aportan más datos debido a la implementación de nuevas herramientas hardware que nos devuelven datos capaces de ser procesados por cualquier aplicación con los permisos necesarios para acceder a ellas. Algunas de éstas herramientas son el GPS, cámara, sensor de luminosidad y giroscopio, aunque hay bastantes más en éste elenco. No obstante, para la utilización de ésta aplicación se utilizará básicamente la cámara y, de forma más o menos habitual, también se habilitarán permisos para la utilización del GPS e Internet.

Es de especial relevancia en este proyecto también la facilidad de implantación en cualquier tipo de dispositivo, por lo que se ha elegido el sistema operativo Android, debido a su auge a nivel mundial y, sobretodo, a nivel del territorio español.

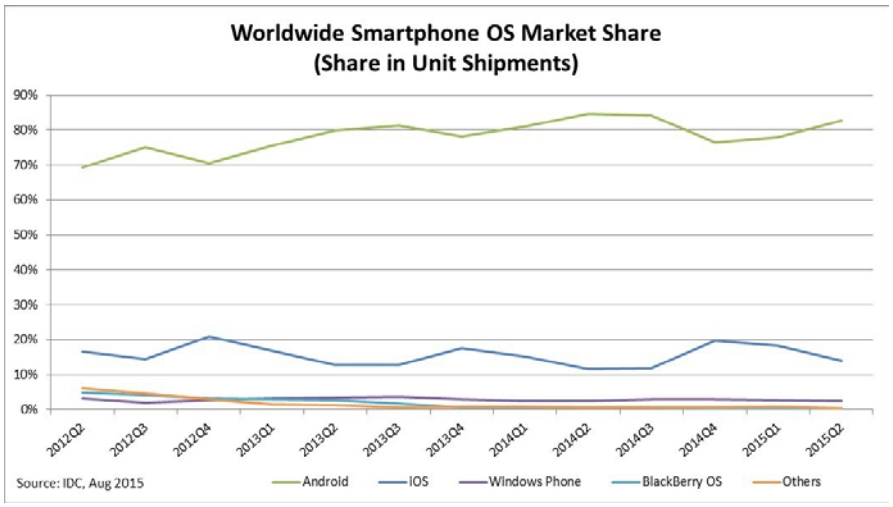


Ilustración 12. Ventas de smartphones según su sistema operativo a nivel mundial [8]

Es de notable destacar también que para desarrollar el proyecto en Android se ha utilizado un entorno de desarrollo proporcionado por NVIDIA, conocido como NVIDIA Tegra Android Development Pack [9], el cual incluye algunos programas y librerías precargadas para el desarrollo de aplicaciones en Android, como puede ser OpenCV, el cual será utilizado para el tratamiento y procesamiento de imagen.

Se ha elegido OpenCV para el tratamiento y procesamiento de imagen debido a que es una librería de código abierto que proporciona un API (Application Programming Interface) para desarrollo en Java/Android y es lo suficientemente potente y amplia como para poder hacer todos los procesos necesarios para reconocer las formas a tratar (círculos, triángulos y cuadrados) y reconocer unas señales de tráfico previamente seleccionadas.

Hay que tener en cuenta que Android también tiene integrado un entorno de bases de datos como es SQLite, el cual será de utilidad a la hora de realizar una captura de datos si ésta fuera necesaria.

Para una mayor especificación y desglose del sistema operativo, entorno de desarrollo y librerías utilizadas se ha de consultar el capítulo 3 de esta memoria.

2.5. Conclusiones del análisis

Después de haber analizado las aplicaciones y sistemas similares capaces de detectar y reconocer señales de tráfico verticales en este capítulo, se ha podido observar que la mayoría ofrecen características muy similares a la aplicación a desarrollar, si bien también complementan algunas características propias como la integración de avisos en paneles propios de los vehículos, como el panel de instrumentos, o también algunos utilizan detección láser, como el coche autónomo de Google, entre otras cosas.

También hay que destacar que las aplicaciones utilizadas no han tenido buenos resultados a la hora de la detección, y las que no han sido utilizadas no se pueden analizar en profundidad por su imposibilidad o dificultad a la hora de obtenerlas.

En el caso de los sistemas implementados en los vehículos, debido a que está en prototipos o en vehículos de gama media-alta o alta, es complicado poder obtener una comparativa objetiva de manera fehaciente, por lo que, mayormente, hay que fiarse de lo que digan las especificaciones del producto y algunos comentarios en revistas y páginas web especializadas.

Asimismo, se ha planteado la posibilidad de realizar una aplicación para detección y reconocimiento de señales y otra que, únicamente, mostrase las señales almacenadas por la

anterior con un posterior procesamiento de los datos obtenidos para no obtener redundancia, pero debido a la carga de autoformación que lleva este proyecto y a la cantidad de tiempo que requiere el procesamiento de imagen, sobretodo en dispositivos móviles, se ha descartado por el momento.

3. SISTEMA OPERATIVO Y ENTORNO DE DESARROLLO

3.1. Introducción

En este capítulo se expondrá el sistema operativo para el que se ha desarrollado la aplicación, haciendo énfasis en el entorno de desarrollo de éste, partiendo del pack de desarrollo que contiene a este entorno y a otras librerías, como a las librerías más importantes, tanto del propio sistema operativo como de las instaladas en el pack.

3.2. Sistema Operativo: Android

Ilustración 13: Logo de Android [14]

Android es un sistema operativo inicialmente pensado para teléfonos móviles. Su núcleo está basado en Linux, lo que favorece que tenga un sistema operativo libre, gratuito y multiplataforma.

El sistema permite programar aplicaciones en una variación de Java llamada Dalvik, que es una máquina virtual utilizada por Android. La DVM (Dalvik Virtual Machine) permite ejecutar aplicaciones programadas en Java, aunque oficialmente no afirma ser una JVM (Java Virtual Machine). Es gracias a Dalvik lo que le permite a Android ser multiplataforma, siendo así de gran implantación en múltiples y diversos dispositivos actuales.

Volviendo a Android, éste es capaz de proporcionar todas las interfaces necesarias para desarrollar aplicaciones que accedan a ciertas funciones del teléfono (como el GPS, las llamadas, la agenda, etc.) de una forma relativamente sencilla en un lenguaje de programación muy conocido como es Java y en lenguaje de marcas extensible como es XML.

Debido a su alta compatibilidad con Java gracias a Dalvik, y a las interfaces que proporciona el propio sistema operativo, se pueden encontrar diversidad de librerías compatibles para su implantación en aplicaciones que utilicen Android, tal es así, que se pueden encontrar librerías como OpenCV y SQLite, entre muchas otras.

También es de destacar la amplia penetración en el mercado de este SO en dispositivos móviles en España a junio de 2015, que se estima del 88,1% como se puede ver en la Ilustración 14, por lo que se presenta como una muy buena opción para desarrollar aplicaciones en estos dispositivos.

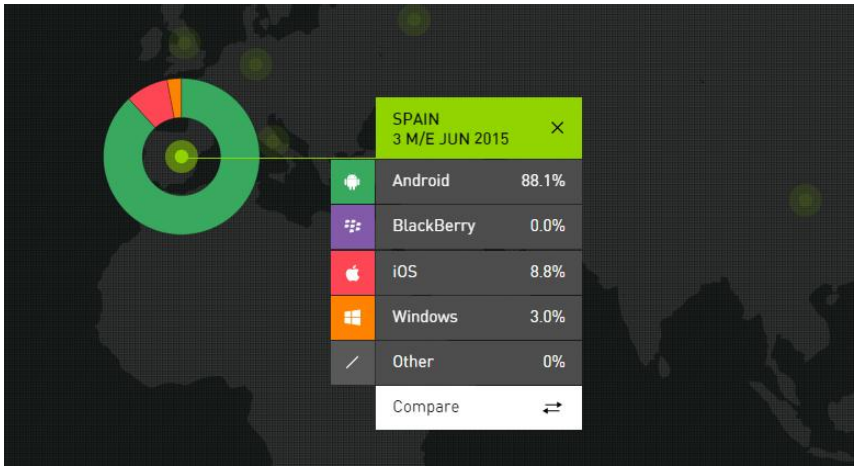


Ilustración 14: Cuota de mercado de dispositivos móviles [15]

3.3. NVIDIA Tegra Android Development Pack

NVIDIA Tegra Android Development Pack (TADP) [9] es un pack de desarrollo de aplicaciones para Android creado por NVIDIA dedicado, sobre todo, a desarrolladores de videojuegos y de aplicaciones con una alta carga de procesamiento de imagen en dispositivos con este sistema operativo, especialmente aquellos que tengan integrados una gráfica de la gama NVIDIA Tegra.

La versión utilizada es la TADP 4.0R1, que es la que estaba más actualizada y disponible a la hora de la realización del proyecto.

Son de destacar para el desarrollo de este trabajo los siguientes elementos del TADP:

- Herramientas utilizadas para el desarrollo de Android:
  - Android SDK r24.0.2
  - Android APIs
  - Android NDK32 and NDK64 r10d

- Android Build Tools r21.1.2
- Android Platform Tools r21
- Android Support Library r21
- Android Support Repository Library r9
- Google USB Driver r11
- JDK 1.7.0_71
- Eclipse 4.3, CDT 8.2.0, ADT 24.0.2
- Apache Ant 1.8.2
- Gradle 2.2.1

- Librerías para Android:
  - OpenCV for Tegra 2.4.8.2

3.4. Eclipse

Ilustración 15. Logo de eclipse [16]

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrolladores. Es utilizada típicamente para desarrollar entornos de desarrollo integrados (IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus.

Eclipse es desarrollado actualmente por la Fundación Eclipse (Eclipse Foundation), una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Este programa, debido su gran capacidad y al instalarse a partir del TADP, incluyendo así el Android SDK (Software Development Kit), es utilizado para el desarrollo de todo el proyecto. Hay que destacar que esde fácil utilización, completo e intuitivo.

3.5. OpenCV

Ilustración 16: Logo de OpenCV [17]

OpenCV es una librería de visión artificial por ordenador de código abierto. Ésta librería está escrita en C y C++ y funciona bajo Linux, Windows y Mac Os X. También existe un desarrollo activo en interfaces para Python, Ruby, Matlab y otros lenguajes como Java.

OpenCV está diseñado para tener alta eficiencia computacional y con un fuerte enfoque para aplicaciones en tiempo real. OpenCV está escrito y optimizado en C y puede aprovechar múltiples procesadores. [18]

Esta librería es utilizada en el proyecto con el fin de procesar las imágenes, aplicarles filtros, algoritmos y descriptores para un correcto procesamiento de las mismas obtenidas por medio de la cámara del dispositivo móvil, gracias a una de las líneas de desarrollo para Java/Android que tiene actualmente.

También es de destacar que ha sido necesaria la instalación de la aplicación *OpenCV Manager* que se encuentra en Google Play, ya que es necesaria para la depuración del código y para poder realizar las conexiones y pruebas necesarias con el dispositivo móvil que tiene Android como sistema operativo. Sin embargo, hay que tener en cuenta que para la instalación de la aplicación final y que ésta pueda estar operativa no es necesaria su instalación.

3.6. SQLite

Ilustración 17: Logo de SQLite [19]

SQLite es un motor de base de datos SQL embebido. A diferencia de la mayoría de bases de datos SQL, SQLite no tiene un servidor de procesos independiente. SQLite lee y escribe directamente a archivos de disco ordinarios. Una base completa de SQL está contenida en un único archivo de disco con varias tablas, índices, triggers y vistas. El formato de archivo de bases de sdatos es multiplataforma (se puede copiar libremente una base de datos entre los sistemas de 32 bits y de 64 bits, o entre arquitecturas big-endian y little-endian). Estas características hacen que SQLite sea una opción popular como formato de archivo para aplicaciones. [19]

Debido a sus características, SQLite está incluido en las librerías propias del sistema operativo de Android en sus versiones más recientes, lo que facilita la creación de una base de datos útil y de fácil utilización para la escritura y lectura de datos entre e intra aplicaciones desarrolladas en este sistema operativo.

4. CONCEPTOS BÁSICOS DE IMAGEN DIGITAL

4.1. Introducción

Debido a la importancia del tratamiento de imagen digital para el procesamiento de las imágenes y la extracción de información de las mismas en este proyecto, es relevante definir y desarrollar brevemente conceptos como píxel, resolución, profundidad y canal de una imagen digital. También, debido a su utilización en este proyecto se procederá a definir conceptos como modelo de color y formato de imagen, dando a conocer y definiendo aquellos utilizados.

4.2. Píxel

El **píxel** es la unidad mínima de representación y visualización de una imagen digital. Su nombre es un acrónimo proveniente del inglés *picture element*, es decir, elemento de imagen.

Ilustración 18. Aumento de imagen para la visualización de los píxeles que la componen

Como se puede ver en la Ilustración 18, se ha ampliado una parte de una señal en la que se ve la parte superior del peatón atravesando un paso de peatones. A la izquierda tenemos la imagen original, y a la derecha la parte de la imagen original marcada con un recuadro en rojo. En la imagen de la derecha, se puede ver que hay una matriz de cuadrados con distintos colores, en la que cada unidad de esa matriz es un píxel, es decir, cada pequeño cuadrado que contiene un único color.

La representación de un píxel viene dada por valores en órdenes de magnitud binarios (2^N) en cada uno de las canales de los que esté compuesta la imagen que se esté visualizando, dependiendo del tipo de profundidad de color por canal y de modo de color utilizado.

4.3. Resolución

La **resolución** de una imagen digital es el grado de detalle que tiene ésta en relación a los píxeles que ésta tenga. Se suele expresar en **ppp** (píxeles por pulgada) o **dpi** (dot per inch) en inglés. Por lo tanto, cuanto más píxeles contenga una imagen por unidad de superficie mayor calidad tendrá.

Ilustración 19. Imagen a 100ppp

Ilustración 20. Imagen a 600ppp

Un ejemplo de la importancia de la resolución se encuentra en las ilustraciones previas a este párrafo. En la Ilustración 19 se puede ver ver una imagen con menor resolución y, por lo tanto, de peor calidad. En la Ilustración 20 se tiene una imagen de mayor resolución y, como se puede apreciar, tiene una mayor calidad de imagen. Es necesario reseñar que, a mayor resolución en un mismo formato, más cantidad de información contiene la imagen, por lo tanto ocupará más en memoria, siempre y cuando tenga el mismo formato de imagen.

4.4. Profundidad del color

Se entiende como **profundidad del color** al número de bits necesarios para codificar y guardar la información de color de cada píxel en una imagen. Cuanto mayor sea la profundidad de color en bits, más cantidad de colores posibles tendrá la imagen y más información tendrá que ser codificada y guardada para su representación.

Debido a su representación en bits, y al ser ésta una medida binaria (0 ó 1), la profundidad de color se mide en órdenes de magnitud binaria (2^N), de ésta forma, teniendo un

solo canal con una profundidad de color de 1 bit seríamos capaces de representar una imagen solamente con color blanco y color negro, siendo las zonas blancas las que más luz tengan y las negras las que muestren más oscuridad.

Ilustración 21. Representación de una imagen en escala monocromática

De manera habitual se utiliza una profundidad de color de 24 bits en modos de color compuestos por tres canales (8 bits de profundidad por canal). Esto es debido a que, de manera general, el ser humano es capaz de percibir 2^{24} colores diferentes.

Ilustración 22. Representación de una imagen en escala RGB

4.5. Canal

Debido a la necesidad de representación en distintos tipos de colores y tonalidades en una imagen digital se ha definido el concepto de canal de imagen. Una imagen está definida por la profundidad de color y por su resolución, de tal manera, cada canal contiene información sobre los píxeles de que está compuesta la imagen y de su codificación de color en bits.

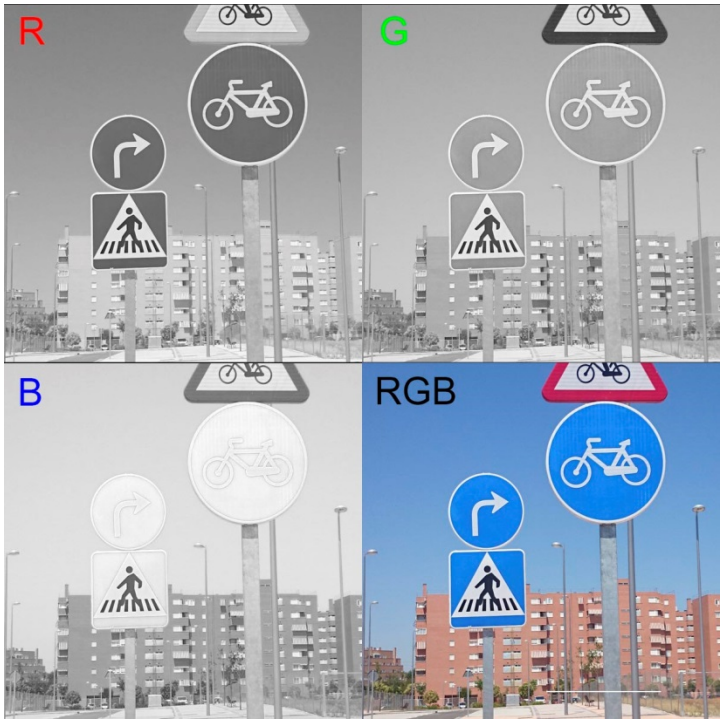


Ilustración 23. Desglose de imagen en RGB por canales y composición final

A modo de ejemplo tenemos la Ilustración 23, en la que podemos ver el modo RGB descompuesto en sus tres canales rojo, verde y azul. Cada canal contiene la cantidad de color que tiene cada píxel en esa imagen y una referencia del canal para indicar, al unir los tres canales, cómo debe quedar la imagen final resultante de su combinación.

4.6. Modelo de color

Como consecuencia de la definición anterior de canal, tenemos los llamados **modelos de color**, cada uno de ellos se define por un modelo matemático abstracto que permite representar los colores en forma numérica, normalmente, superponiendo varios canales.

Los modelos de color utilizados en el proyecto son los siguientes:

- **Monocromático**. Se corresponde con un canal y una profundidad de color de 1 bit. La imagen está formada por píxeles blancos o negros.
- **Escala de grises**. Se utiliza un solo canal y permite 256 (2^8) variaciones de color entre el blanco y negro puros.
- **RGB**. Compuesto por tres canales correspondientes a un color primario, a saber: *Red* (rojo), *Green* (verde) y *Blue* (azul). Asigna un valor de intensidad a cada color que suele oscilar entre 0 y 255.
- RGBA. Es un modo de color RGB con un canal Alpha añadido, es decir, está compuesto por cuatro canales. Este cuarto canal es utilizado para indicar la transparencia de la imagen.

4.7. Formato de imagen

El **formato de imagen** o de archivo se refiere a la estructura de datos en que se guardará un fichero. Cada tipo de codificación lleva asociada una estructura de datos diferente, aunque todas ellas sirven para representar imágenes. En el proyecto se utilizarán los siguientes formatos de imagen:

- **BMP** (*Bitmap* = Mapa de bits). La imagen se forma mediante una matriz de píxeles. Este formato no sufre pérdida de calidad, lo que supone archivos de imagen voluminosos.
- **JPG-JPEG** (*Joint Photographic Experts Group*). Es uno de los formatos más utilizados para guardar imágenes por las cámaras de sistemas móviles, debido a que es fácil de leer en cualquier tipo de dispositivo y para subirlas y/o compartirlas en entornos web o entre dispositivos.
- **PNG** (*Portable Network Graphics*). Es utilizado por Android para ser utilizado en sus aplicaciones, ya sea en iconos, imágenes dentro de la aplicación, etc. debido a que tiene una capa alpha de transparencia que permite mostrar imágenes superpuestas, formas peculiares en las imágenes mostrando el contenido inferior sobre todo.

5. FILTRADO, DETECCIÓN Y RECONOCIMIENTO DE IMAGEN

5.1. Introducción

Para realizar procesamiento de imagen, sobre el que se basa este proyecto, es necesaria la aplicación de ciertos filtros de imagen, con el fin de poder ir tratando las imágenes obtenidas y sacar la mayor información útil para los distintos casos prácticos que se requieran.

Además de los filtros, también se hace necesaria la utilización de algoritmos de comparación de imagen para poder reconocer las imágenes obtenidas y así encontrar coincidencias con aquellas precargadas en la aplicación.

5.2. Desenfoque gaussiano (Gaussian blur)

Como parte fundamental básica de un procesamiento de imagen digital inicial se necesita disminuir la cantidad de ruido de una imagen para poder tratarla mejor y tener que procesar menor cantidad de información que no es relevante en la misma. Para ello se va a utilizar la difuminación o desenfoque gaussiano (Gaussian blur) [20], que se explicará en los siguientes párrafos, no sin antes introducir la transformada gaussiana.

En base a lo expuesto en el párrafo anterior, es realmente sorprendente el número de operaciones de procesamiento de imágenes pueden ser expresadas por una simple ecuación:

$$\vec{\hat{v}}_i = \sum_j e^{-\frac{|\vec{p}_i - \vec{p}_j|^2}{2}} \vec{v}_j$$

Ecuación 1. Transformada gaussiana

Esta fórmula (Ecuación 1) es la transformada gaussiana, en la cual los *valores* de salida $\vec{\hat{v}}_i$ son ponderados por la suma de los *valores* de entrada $\vec{v}_j$, con los pesos dados por una gaussiana en la distancia asociada entre dos *posiciones* $\vec{p}_i$ y $\vec{p}_j$. En pocas palabras, esta ecuación combina *valores* que tienen *posiciones* similares.

Para el procesamiento de imagen, nuestros valores serán casi siempre colores de píxeles, así que $\vec{v}_i$ representa el color del pixel i. Se utiliza una representación homogénea para el color, así que el sumatorio de la *Ecuación 1* realiza un promedio ponderado. Esto es descrito como un *filtrado gausiano*. Luego un píxel con componentes rojo, verde y azul tiene un valor de entrada $[r, g, b, 1]$, y un valor de salida $[a, b, c, d]$ debería ser entendido como el color $\left[\frac{a}{d}, \frac{b}{d}, \frac{c}{d}\right]$. Las posiciones asociadas a estos valores variarán dependiendo a la tarea en cuestión.

Teniendo en cuenta lo anteriormente expuesto y, sobretodo, lo citado en el párrafo anterior sobre el filtrado gaussiano, el caso particular utilizado es el de *desenfoque gaussiano* o *Gaussian blur*. Este filtro se caracteriza por $\vec{p}_i = \left(\frac{x_i}{\sigma}, \frac{y_i}{\sigma}\right)$ y $\vec{v}_i = (r_i, g_i, b_i, 1)$, siendo x e y las coordenadas bidimensionales en la imagen y siendo σ la desviación típica aplicada.

Por ejemplo, si se fijara el vector de posición del píxel i para que sea (x,y) la localización del píxel con la imagen, entonces la Ecuación 1 expresaría un desenfoque gaussiano de desviación 1. Si se deseara realizar un gran desenfoque, se podría modificar la ecuación, pero sería más conveniente tratar de disminuir la escala de los vectores de posición.

Ilustración 24. Fotografía de carretera

Ilustración 25. Fotografía de carretera con desenfoque gaussiano

En este tipo de filtro, en el cual los valores son los colores del píxel y la posición de cada píxel en su localización espacial bidimensional en la imagen. Estos valores se mezclan con aquellos que tienen posiciones muy cercanas en la imagen, mezclando así sus colores y produciendo difuminación en imagen. Éste comportamiento sería similar al que tendría un miope en largas distancias o a un hipermétrope en distancias cortas.

5.3. Detector de bordes Canny (Canny Edge Detector)

Una medida previa a la detección de cualquier tipo de forma es poder obtener su contorno, de tal manera que así se pueda obtener información sobre una forma y determinar su tipo, ya sea poligonal o no. Por eso, y con este propósito, necesitamos un filtro que contenga un algoritmo de detección de bordes para poder procesar la imagen y obtener sus formas.

Dicho lo anterior, se utilizará el filtro de detección de bordes Canny, está basado en el algoritmo propuesto por John F. Canny en 1986 [21]. A veces también conocido como el **detector óptimo**, éste algoritmo pretende satisfacer tres criterios principales [22]:

1. **Baja tasa de error**. Todos los bordes deben de ser encontrados, y no deben de ser respuestas espurias. Esto significa que, los bordes detectados deben estar lo más cerca posible de los bordes verdaderos.
2. **Los puntos de los bordes deben de estar bien ubicados.** La ubicación de los puntos de los bordes debe estar tan cerca como sea posible de los bordes reales. Esto significa que la distancia entre un punto marcado como un borde por el detector y el centro del borde real debe ser la mínima.
3. **Obtención de un solo punto en el borde.** El detector debe retornar un único punto por cada punto de borde real. Esto significa que el número de máximos locales alrededor del borde real debe ser el mínimo, así pues el detector no debería identificar múltiples píxeles de borde cuando solo existe un único punto en el borde.

Para conseguir los criterios anteriores, el filtro de detección de bordes Canny se realiza con los siguientes pasos básicos:

1. **Reducción de ruido**. El detector de bordes es sensible al ruido en la imagen, por lo que el primer paso es eliminar ruido en la imagen con un filtro gaussiano, como puede ser el de desenfoque (Gaussian blur).
2. **Encontrar la intensidad de gradiente en la imagen**. La imagen desenfocada es filtrada posteriormente por un operador Sobel tanto en dirección horizontal como en vertical para obtener la primera derivada en dirección horizontal (G_x) y vertical (G_y). A partir de estas derivadas podemos encontrar el gradiente del borde y la dirección de cada píxel como sigue:

$$Gradiente\_del\_borde(G) = \sqrt{G_x^2 + G_y^2}$$
$$Ángulo\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Ecuación 2. Gradiente del borde y ángulo en algoritmo Canny

   La dirección del gradiente es siempre perpendicular a la del borde. Se redondea a uno de los cuatro ángulos que representan verticalidad, horizontalidad o a las dos direcciones diagonales.
3. **Supresión de puntos que no se encuentren en un máximo**. Después de obtener la magnitud y dirección del gradiente, se realiza un análisis completo de la imagen para eliminar los píxeles no deseados que pueden no constituir el borde. Para ello se comprueba si cada píxel es un máximo local en su cercanía teniendo en cuenta la dirección del gradiente, tal y como se muestra en la siguiente imagen:

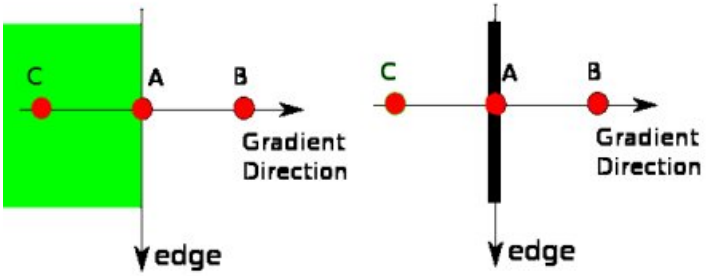


Ilustración 26. Supresión de puntos que no están en un máximo [23]

El punto A está en el borde (dirección vertical). La dirección del gradiente está en la normal con respecto al borde. Los puntos B y C están en la dirección del gradiente. Sin embargo, el punto A está marcado entre los puntos B y C para ver si es un máximo local. Si es así, se considera para la siguiente etapa, de no ser así, se suprime (se pone a cero).

4. **Umbral de histéresis.** Este estado decide qué bordes lo son realmente y cuáles no. Para esto, se necesitan dos valores umbrales, minVal (valor mínimo) y maxVal (valor máximo). Los bordes con intensidad de gradiente más cercanos a maxVal tienen más posibilidad de ser bordes verdaderos que los que se encuentran más cerca de minVal, así que estos últimos son descartados. Aquellos que se encuentran entre estos dos umbrales se clasifican como bordes o no dependiendo de su conectividad. Si están conectados a píxeles de bordes reales son considerados como parte de los bordes, si no son descartados.

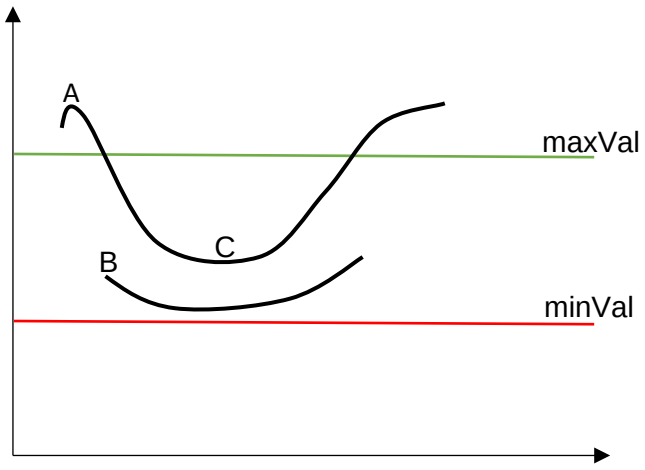


Ilustración 27. Decisión sobre puntos en relacionados con bordes reales

El borde A está sobre maxVal, así que está considerado como un borde real. Sin embargo, C está bajo maxVal, pero conectado al borde A, así que es un borde válido y se podrá coger la curva completa. Pero B, está sobre minVal y está en la

misma región que C, pero no está conectado a ningún borde verdadero, así que es descartado. Es importante seleccionar el minVal y el maxVal de acuerdo a obtener un buen resultado.

Este estado elimina los píxeles pequeños de ruido asumiendo que los bordes son largas líneas.

Ilustración 28. Fotografía de carretera

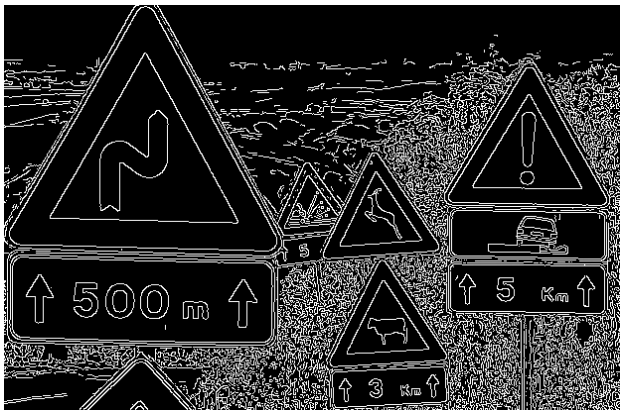


Ilustración 29. Fotografía de carretera procesada con algoritmo Canny

Ilustración 30. Fotografía de carretera desenfocada procesada con algoritmo Canny

En las imágenes anteriores se puede ver el funcionamiento del algoritmo de detección de bordes Canny. En la Ilustración 28 podemos ver una fotografía de una señalización real en una carretera, y en la Ilustración 29 se puede ver esa misma fotografía procesada por éste algoritmo. También podemos ver en la Ilustración 30 cómo, si aplicamos un desenfoque gaussiano, la imagen obtenida para procesar posteriormente las formas geométricas es mucho más sencilla y contiene mucha menos información irrelevante, por lo que posteriormente será mucho menos costosa procesarla y extraer información relevante de la misma.

5.4. Transformada de Hough (Hough Transform)

Debido a la necesidad de detectar unas formas determinadas, es necesario obtener un algoritmo que reconozca ciertos patrones para extraer aquellos que más nos interesen.

La **Transformada de Hough** cumple el cometido expuesto anteriormente, ya que es una técnica para la detección de patrones de figuras en imágenes digitales. Se suele utilizar para encontrar figuras que puedan expresarse mediante técnicas matemáticas, tales como rectas, circunferencias o elipses. Inicialmente propuesta por Paul Hough en 1962 [24], se aplicaba únicamente a rectas en una imagen. Posteriormente, gracias al trabajo de Richard Duda y Peter Hard en 1972 [25] se obtuvo la *"Transformada de Hough Generalizada",* que es la que se utiliza hoy en día, ya que permite la utilización para distintas formas con cierta geometría. Es utilizada actualmente en procesamiento de imagen digital gracias a Dana H. Ballard en 1981 [26].

Esta transformada actúa suponiendo que, para n puntos de una imagen, se desean encontrar subconjuntos de éstos que residan en líneas rectas.

Si se considera un punto (x_i, y_i) y la ecuación general de una recta en forma explícita $y_i = ax_i + b$. A través de (x_i, y_i) pasan un número infinito de líneas, pero todas ellas satisfacen la ecuación general de una recta para diversos valores de a y b. Sin embargo, escribiendo esta ecuación como $b = -x_i a + y_i$ y considerando el $plano\ ab$ (también denominado *espacio de parámetros*) se obtiene la ecuación de una única línea para un par determinado (x_i, y_i). Además, un segundo punto (x_j, y_j) también tiene una línea en el espacio de parámetros asociado con él, y esta línea corta a la línea asociada con (x_i, y_i) en (a', b'), donde a' es la pendiente y b' la ordenada en el origen de la línea que contiene a (x_i, y_i) y a (x_j, y_j) en el $espacio\ xy$. De hecho, todos los puntos contenidos en esta línea tienen líneas en el espacio de parámetros que corta a (a', b').

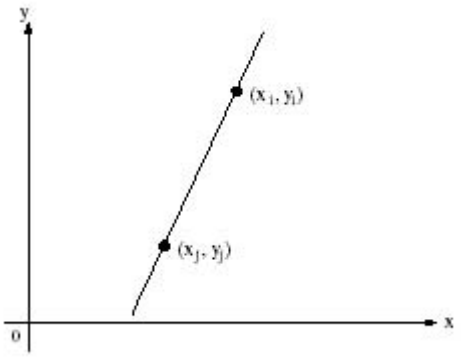


Ilustración 31. Plano xy [27]

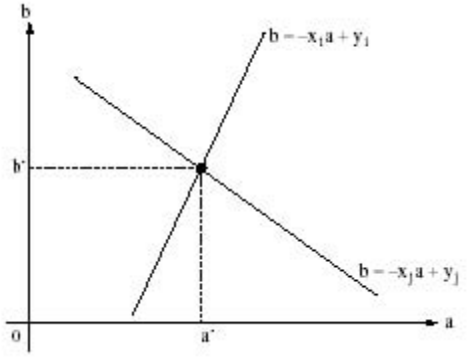


Ilustración 32. Espacio de parámetros [27]

La parte interesante del cálculo de la transformada de Hough surge de la subdivisión del espacio de parámetros en las denominadas células acumulador donde (a_{max}, a_{min}) y (b_{max}, b_{min}) son los rangos esperados de los valores de la pendiente y ordenada. La célula de coordenadas (i, j), con valor de acumulador $A(i, j)$, corresponde al cuadrado asociado con las coordenadas del espacio parámetro (a_i, b_j). Inicialmente estas células están puestas a cero. Después, para cada punto (x_k, y_k) del plano imagen, se fija el parámetro a igual a cada uno de los valores permitidos de subdivisión sobre el $eje\ x$ y se resuelve para el b correspondiente utilizando la ecuación $b = -x_k a + y_k$. Las b resultantes se redondean después al valor más próximo permitido del $eje\ x$. Si una elección de a_p resulta ser la solución b_q, se fija $A(p, q) = A(p, q) + 1$. Al final de este procedimiento, un valor de M en $A(i, j)$ corresponde a M puntos del $plano\ xy$ situados en la línea $y = a_i x + b_j$.

La precisión de la colinearidad de estos puntos está determinada por el número de subdivisiones del $plano\ ab$.

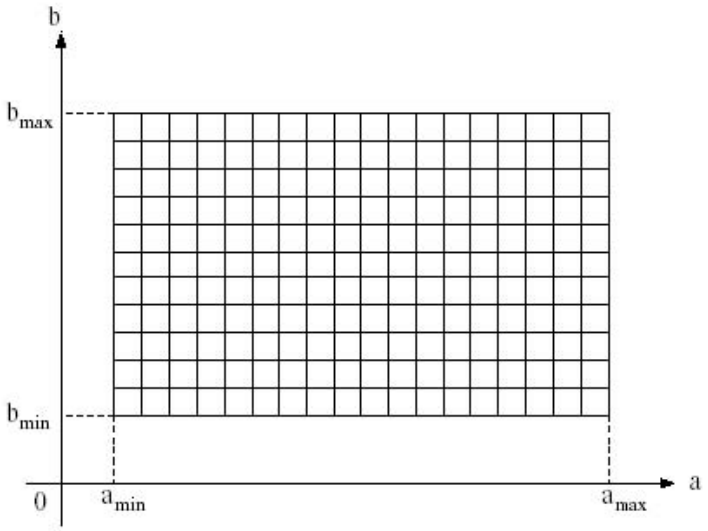


Ilustración 33.Cuantificación del plano de parámetros para utilizar la transformada de Hough [27]

Un problema que aparece al utilizar la ecuación $y = ax + b$ para representar una línea es que tanto la pendiente como la ordenada en el origen tienden al infinito cuando la línea se acerca a la vertical. Una forma de evitar esta dificultad es utilizar la representación normal de una recta:

$$x\cos\theta + y\sin\theta = \rho$$

Ecuación 3. Representación normal de una recta

El empleo de esta representación en la construcción en una tabla de acumuladores es idéntico al método expuesto para la representación punto-pendiente. Sin embargo, en vez de líneas rectas, los lugares geométricos son curvas sinusoidales en el $plano\ \rho\theta$. Como antes, M puntos colineales situados sobre una línea $x\cos\theta_j + y\sin\theta_j = \rho_i$ producen M curvas sinusoidales que cortan a (ρ_i, θ_j) en el espacio de parámetros. Incrementando θ y calculando ρ, obtendremos M entradas en el acumulador $A(i, j)$ correspondiente al par (ρ_i, θ_j).

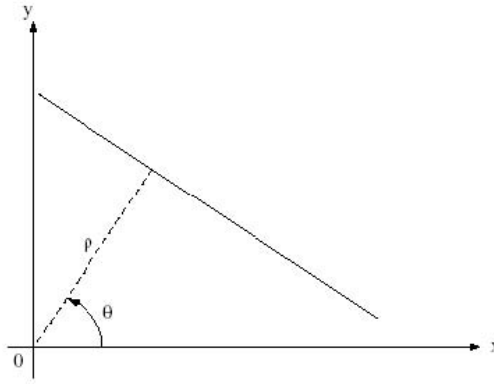


Ilustración 34. Representación normal de una línea [27]

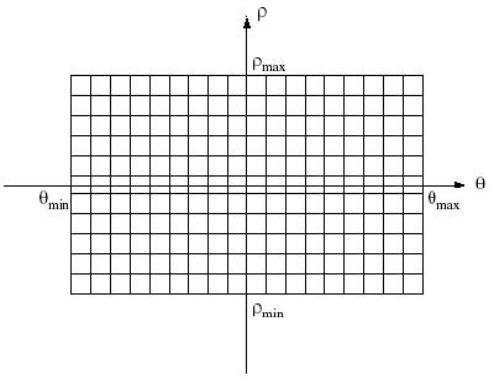


Ilustración 35. Cuantificación del plano ρθ en células [27]

El rango para el ángulo θ es $\pm$90º, medido con respecto al eje de abscisas. Se permiten valores negativos de θ para rectas por detrás del origen de coordenadas del $plano\ xy$. Por ejemplo, una recta horizontal corresponde a un valor de $\theta = 0º$ y un valor de ρ igual a la ordenada al origen, mientras que una recta vertical corresponde a un valor de $\theta = 90º$ y un valor de ρ igual a la abscisa en el origen.

Aunque se ha explicado el procedimiento para rectas, la transformada de Hough también es aplicable a cualquier función de la forma $g(v, c) = 0$ donde v es un vector de coordenadas y c es un vector de coeficientes. Por ejemplo, los puntos que caen en el círculo $(x - c_1)^2 + (y - c_2)^2 = {c_3}^2$ se pueden detectar empleando también esta transformada. En este caso tenemos tres parámetros $(c1, c2, c3)$, lo que dará lugar a un espacio de parámetros

de tres dimensiones, con celdas con forma de cubo y acumuladores de la forma $A(i, j, k)$. El procedimiento en este caso es para cada punto del $plano\ xy$, para cada c_1 y para cada c_2, calcular el valor de c_3 y actualizar el acumulador correspondiente a (c_1, c_2, c_3). La complejidad de la Transformada de Hough es claramente dependiente del tamaño del espacio de parámetros.

Ilustración 36. Muestra de detección de círculos Hough [17]

En la Ilustración 36 se muestra se puede ver el resultado de aplicar el algoritmo de Transformada de Hough para la detección de círculos. Como se puede ver, se ha detectado tanto el círculo, mostrando su circunferencia en rojo, como el centro del mismo mostrado en verde.

5.5. Descriptor ORB

La necesidad de comparar las imágenes para poder identificarlas ha hecho buscar algún algoritmo o algoritmos con los que poder hacerlo. Así pues, para la comparación utilizada para el reconocimiento de imágenes se ha optado por utilizar descriptores, más concretamente, el descriptor ORB.

ORB (*Oriented FAST and Rotated BRIEF)* es un rápido y robusto descriptor binario de características locales, que fue presentado inicialmente por Ethan Rublee et al. en 2011 [28] y puede ser utilizado en tareas de visión por ordenador, tales como reconocimiento de objetos o reconstrucción en 3D. Está basado en el detector de *keypoints* FAST (Features from Accelerated Segment Test) [29] y en el descriptor visual BRIEF (Binary Robust Independent Elementary Features) [30]. Su objetivo es proporcionar una alterativa a SIFT (Scale Invariant Feature Transform) [31] y SURF (Speeded-Up Robust Features) [32] en cuanto a rapidez y eficiencia en el cálculo, la búsqueda de pares de muestras y sobre todo, a su licencia gratuita.

Para entender mejor ORB es necesario conocer las partes básicas de las que se compone un descriptor binario, que son:

1. **Un patrón de muestreo**, en el cual poder muestrear los puntos en la región acotada por el descriptor.
2. **Compensación de orientación** mediante algún mecanismo que permita medir la orientación de algún punto significativo y girarlo para compensar los cambios de rotación.
3. **Pares de muestras**, para poder comparar a la hora de construir el descriptor final.

Hay que tener en cuenta que para construir la cadena binaria que representa una región acotada a partir de un *keypoint* se necesitan repasar todos los pares de muestras y para cada par $(p1, p2)$, si la intensidad en el punto $p1$ es mayor que la intensidad en el punto $p2$, se registra un 1 en la cadena binaria o un 0 en caso opuesto.

El descriptor ORB utiliza como base el descriptor BRIEF, aunque hay dos diferencias importantes entre ellos:

1. ORB utiliza un mecanismo de **compensación de orientación**, realizando una rotación invariante.
2. ORB aprende los **pares de muestras óptimos**, mientras que BRIEF utiliza pares de muestras elegidos al azar.

El método de compensación de orientación utilizado trata de una simple medida de la orientación de los *keypoints* obtenidos mediante el detector FAST denominada **intensidad de centroide** (*intensity centroid*). En primer lugar, se definen los momentos de un área como:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y)$$

Ecuación 4. Momentos en un área de una imagen

Con estos momentos podemos encontrar el "centro de masas" de un área como

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right)$$

Ecuación 5. Centro de masas

Se puede construir el vector de orientación desde el centro de un keypoint O, al centroide, $\overrightarrow{OC}$. La orientación del área se puede hallar fácilmente mediante:

$$\theta = atan2(m_{01}, m_{10})$$

Ecuación 6. Ángulo de orientación de un área

En la siguiente ilustración se puede ver de manera simplificada la explicación del método.

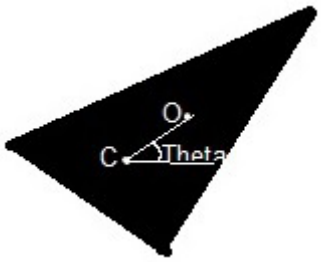


Ilustración 37. Explicación gráfica de la orientación de un área en una imagen [33]

Así pues, ORB lo que hace es "dirigir" BRIEF de acuerdo a la orientación θ anteriormente obtenida mediante los keypoints. Para cualquier conjunto de n pruebas binarias en la ubicación (x_i, y_j), se define una matriz $2xn$ denominada S y que contiene las coordenadas de esos píxeles. Posteriormente, se utiliza la rotación del área θ para rotar la matriz S y así obtener una versión rotada de ésta denominada S_θ.

Para el aprendizaje de pares de muestras óptimo se ha de tener en cuenta una propiedad importante de BRIEF, en la que cada bit tiene como características tener una gran varianza y una media cercana a 0,5. Sin embargo, una vez orientado gracias a la dirección de los keypoints obtenida previamente, pierde sus propiedades y se hace más distribuido. Una gran varianza hace a una característica más discriminativa ya que responde diferencialmente a las entradas. Otra propiedad deseable es tener las pruebas incorreladas, así cada prueba contribuirá al resultado. Para resolver todo esto, ORB ejecuta una búsqueda exhaustiva entre todas las posibles pruebas binarias para encontrar aquellas que tengan una alta varianza a la vez que una media cercana a 0,5, además de ser incorreladas. El resultado de esto se ha denominado rBRIEF.

Como ejemplo de qué se entiende por keypoints tenemos la Ilustración 38 de una señal de tráfico, concretamente la de vía reservada para ciclos, en la cual se pueden ver los keypoints, que se podría decir son los puntos característicos de la señal, eso sí, obtenidos mediante el algoritmo que utiliza ORB.

Ilustración 38. Señal de tráfico con keypoints identificados en azul

Posteriormente, para ver la capacidad comparadora del descriptor ORB, si se procesan los *keypoints* de ambas imágenes se encuentran pares de muestras enlazados la imagen anterior junto con otra real, enlazándose así sus *keypoints* y marcando las coincidencias, obteniendo así una relación entre las imágenes.

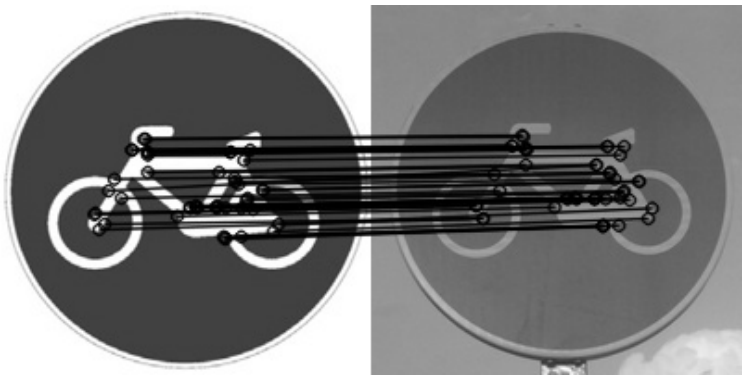


Ilustración 39. Señales de tráfico comparadas con pares de coincidencias encontrados y enlazados

Las imágenes de la Ilustración 38 y la Ilustración 39 han sido procesadas en escala de grises para aumentar su precisión y disminuir el tiempo de procesamiento.

5.6. Clasificador en cascada (Cascade Classifier)

La necesidad de comparar las imágenes para poder identificarlas ha hecho inicialmente fue pensada para ser realizada mediante el *algoritmo de detección en cascada* propuesto por Paul Viola y Michael Jones en 2001 [34].

La contribución más importante del trabajo de Viola y Jones fue la definición de la cascada *"Attentional*". Éste un árbol de decisión degenerado donde, en cada fase, un detector es entrenado para detectar casi todos los objetos de interés mientras rechaza a los que no lo son.

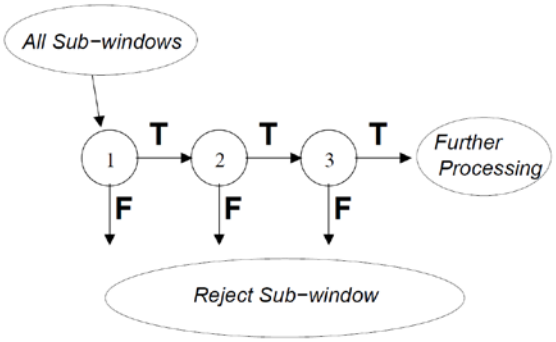


Ilustración 40. Representación esquemática de la detección en cascada [34]

Los sistemas de detección deben cumplir fuertes restricciones, tanto en tasa de aciertos como de fallos. Si se entrena un detector simple con estas restricciones, el número de hipótesis que el método de *boosting* debe combinar para obtener el resultado es enorme. Utilizando la estructura en cascada, la restricción de la tasa de falsas alertas es compartida junto con las alertas falsas de la cascada de detectores. Lo mismo puede ser aplicado para el porcentaje de aciertos.

Cada fase analiza sólo a los objetos aceptados en las fases previas, y los no-objetos son analizados sólo hasta que son rechazados por un detector. El número de clasificadores aplicados se reduce exponencialmente debido a la arquitectura de la cascada. Se utiliza AdaBoost para aprender cada nivel de la cascada y cambiar los objetos rechazados por no-objetos que las fases entrenadas previamente clasifiquen como correctos.

Así pues, después de varios procesos y de aplicar diversas técnicas se genera un fichero donde se contiene toda la información relativa a los objetos aceptados, que servirá como identificador para las señales.

Debido a que no se ha seguido esta línea de investigación se ha decidido únicamente explicar brevemente el funcionamiento de éste procedimiento en esta memoria.

6. ANÁLISIS DE LA APLICACIÓN

6.1. Introducción

En este capítulo se presentará una descripción general del sistema, donde se tratará más en detalle el producto. Se tratará la perspectiva y funciones del sistema, así como las características de los usuarios, interacción con otros sistemas, restricciones y suposiciones y dependencias, tal y como figura en el quinto apartado del estándar IEEE 830 [35].

A continuación, se detallará la especificación de requisitos recogida para el desarrollo de la aplicación y los casos de uso ligados a los requisitos citados.

6.2. Descripción general del producto

Tomando como referencia el estándar IEEE 830 [35], en esta sección se describen todos aquellos factores que afectan al producto y a sus requisitos.

6.2.1. Perspectiva del producto

Partiendo de la base de que la detección de señales de tráfico es un concepto relativamente nuevo, sobretodo en dispositivos móviles, y que las librerías utilizadas para su procesamiento se encuentran aún en fase de desarrollo o en una fase beta permanente, hay que tener en cuenta también las limitaciones tanto del hardware como las del sistema operativo Android utilizados. Dicho esto, se pretende que el producto sea compatible con cualquier dispositivo Android superior a la versión 4.0 y que utilice la menor cantidad de recursos posible para el procesamiento de la imagen y la detección de las señales.

Siguiendo con lo anterior, el catálogo de productos que se pretende ofertar al usuario final es el siguiente:

- **DSTr**. Aplicación creada para plataformas con sistema operativo Android, cuyo cometido es la detección y reconocimiento de señales de tráfico en tiempo real, dando la posibilidad de almacenarlas en una base de datos con un identificador de la señal y su geolocalización. También es posible la exportación de la base de datos para obtener los datos obtenidos de las señales detectadas.

6.2.2. Funciones del producto

En esta sección se van a definir las principales funcionalidades del producto. El sistema se puede dividir en tres grandes bloques, dependiendo de su funcionalidad:

- **Detección y reconocimiento de señales**. Este bloque conceptual contiene las funcionalidades del sistema dedicadas a la detección de la señalización vertical mediante la extracción de las mismas por medio del análisis de las imágenes extraídas por la cámara del dispositivo móvil.
    - **Detección y reconocimiento de señales circulares**. Función encargada de detectar, reconocer y almacenar en la base de datos con datos GPS las señales verticales de forma circular si tienen cierta similitud con señales precargadas en la aplicación.
    - **Detección y reconocimiento de señales triangulares y cuadradas**. Función encargada de detectar, reconocer y almacenar en la base de datos con datos GPS las señales verticales de forma triangular y cuadrada si tienen cierta similitud con señales precargadas en la aplicación.
    - **Calibrar cámara**. Es la función encargada de mostrarnos la imagen real de la escena, para poder calibrar la cámara. Se encuentra dentro de este bloque ya que es parte esencial para la detección tener una imagen previa global de lo que se intenta procesar.
- **Gestionar base de datos.** Bloque conceptual de la aplicación encargada de gestionar la base de datos donde son almacenados los datos extraídos por la función de detección de señales.
    - **Visualizar la base de datos.** Función encargada de mostrarnos los registros de la base de datos, es decir, las señales reconocidas con su geolocalización (latitud y longitud).
    - **Limpiar base de datos.** Función encargada de limpiar los registros guardados en la base de datos.
    - **Exportar base de datos.** Función encargada de exportar la base de datos de la aplicación al almacenamiento externo del dispositivo.

6.2.3. Características de los usuarios

Este proyecto tiene unos perfiles bien diferenciados, no obstante, algunos de los usuarios finales pueden ser de uno o más tipos. De este modo, se definen distintos tipos de perfiles, con el fin de definir mejor a los tipos de usuario.

El primer perfil se corresponde con aquellos usuarios que utilicen la aplicación con fines de obtención de información de la señalización vertical de poblaciones y carreteras españolas para la utilización de los datos extraídos con distintos fines.

El segundo perfil corresponde con los usuarios que utilicen la aplicación con el fin de obtener un aviso previo a la señalización ya existente en la vía con el fin de mejorar la

conducción, reducir el consumo de combustible y disminuir los accidentes y las infracciones de tráfico.

6.2.4. Restricciones

Las restricciones del sistema se describirán con detalle en la sección 6.4.2 del presente documento.

6.2.5. Suposiciones y dependencias

Existen dependencias directas de la aplicación para su funcionamiento, en concreto, es necesario el siguiente hardware:

- **Cámara.** Para la detección de figuras es necesario obtener las imágenes capturadas por la cámara, de no existir ésta o no poder obtener las imágenes, la aplicación no funcionará.

A parte de la cámara, no existen dependencias directas de la aplicación para su funcionamiento. No obstante, para la utilización de algunas de las funcionalidades es necesaria, aunque no imprescindible, la utilización de los siguientes servicios:

- **GPS**. En caso de no estar este servicio disponible, no se guardará la geolocalización de las señales reconocidas, siempre y cuando Internet no esté también disponible.
- **Internet**. En caso de no estar este servicio disponible, no se guardará la geolocalización de las señales reconocidas, siempre y cuando la señal GPS no esté también disponible.

6.3. Casos de uso

En este apartado se expondrán los diagramas de casos de uso en función de los actores que utilicen el sistema, así como su especificación formal en formato tabla.

6.3.1. Diagramas de uso

Los casos de uso sirven para identificar la relación existente entre el actor (usuario que utilice la aplicación) y el software (la aplicación). Para mayor claridad de la utilización de la

aplicación, se expone el siguiente diagrama de uso de un usuario que utilice todas las funcionalidades.

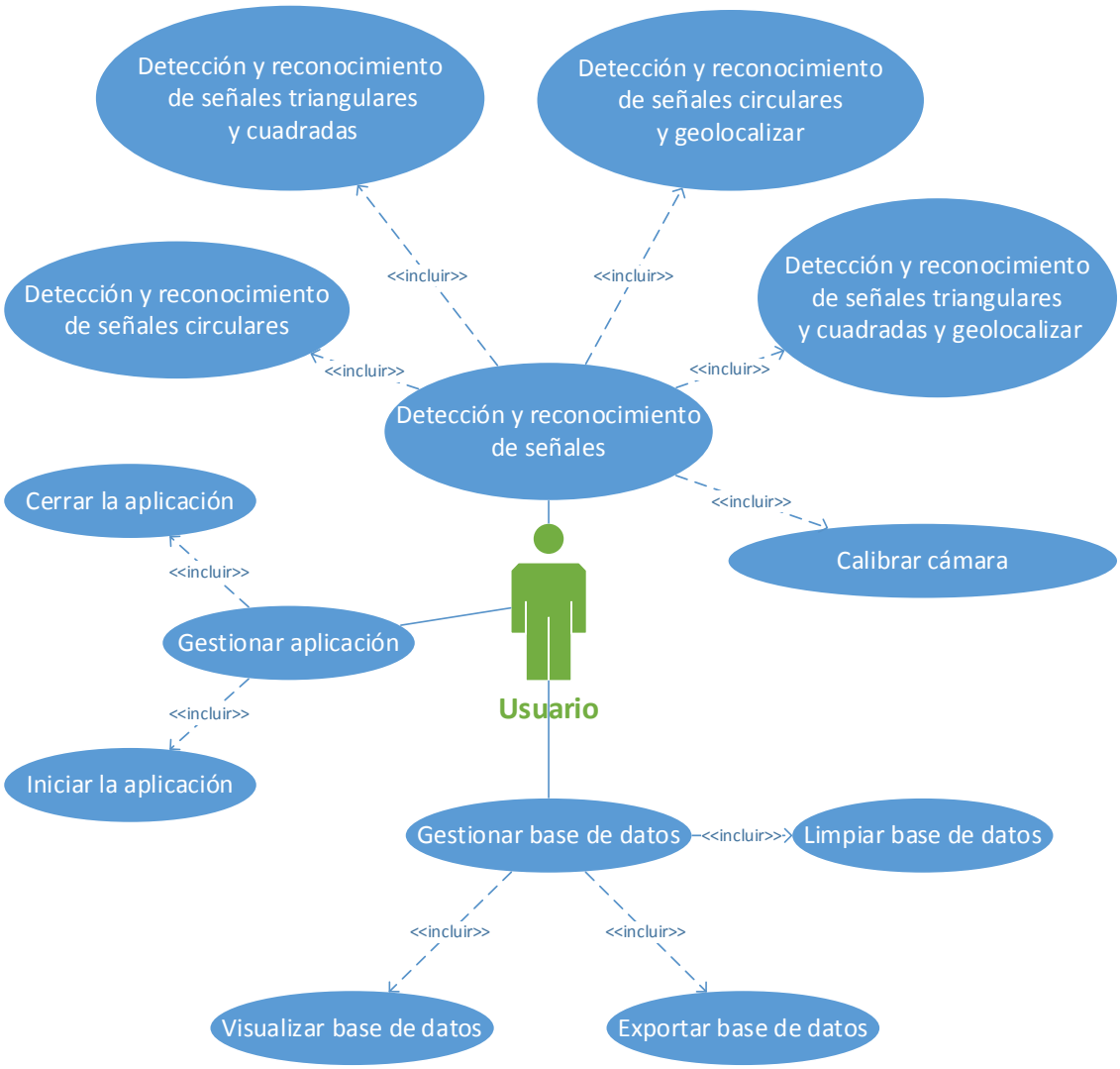


Ilustración 41. Diagrama de casos de uso de la aplicación

En la próxima sección se procederá a especificar formalmente los casos de uso.

6.3.2. Especificación detallada de los casos de uso

En este apartado se describirá la especificación la especificación de los casos de uso.

El ejemplo de tabla que se va a seguir para describir los casos de uso es el siguiente:

Identificador	*CU-XX*	**Nombre**	*Nombre identificativo*
Fuente	*Requisitos del que proviene el caso de uso*		
Actores	*Usuarios que intervienen*		
Objetivo	*Objetivo del caso de uso*		
Precondiciones	*Condiciones anteriores para que se cumpla el caso de uso*		
Postcondiciones	*Condiciones finales tras realizarse el caso de uso*		
Escenario básico	*Escenario básico de caso de uso*		

Tabla 2. Ejemplo de caso de uso

6.3.2.1. Especificación detallada de los casos del usuario

Identificador	*CU-01*	**Nombre**	*Inicio de aplicación*
Fuente	*RSF-01*		
Actores	*Usuario*		
Objetivo	*Iniciar la aplicación*		
Precondiciones	1. *No tener la aplicación abierta*		
Postcondiciones	1. *Tener abierta la aplicación*		
Escenario básico	1. *Acceder al sistema operativo Android* 2. *Pulsar sobre el icono con nombre "Detector de Señales de Tráfico"*		

Tabla 3. Caso de uso 1

Identificador	*CU-02*	**Nombre**	*Cierre de aplicación*
Fuente	*RSF-02*		
Actores	*Usuario*		
Objetivo	*Cerrar la aplicación*		
Precondiciones	1. *Tener la aplicación abierta*		
Postcondiciones	1. *Cerrar la aplicación*		
Escenario básico	1. *Estar dentro de la aplicación* 2. *Pulsar sobre el botón de "Atrás" del sistema operativo Android*		

Tabla 4. Caso de uso 2

Identificador	*CU-03*	**Nombre**	*Detectar y reconocer señales circulares*
Fuente	*CU-01, RSF-03, RSD-01*		
Actores	*Usuario*		
Objetivo	*Detectar señales de tráfico de forma circular*		
Precondiciones	1. *Tener la aplicación abierta* 2. *Tener la cámara habilitada*		
Postcondiciones	1. *Mostrar por pantalla las señales de tráfico circulares reconocidas.*		
Escenario básico	1. *Estar dentro de la aplicación* 2. *Pulsar sobre la opción "Detectar círculos"*		

Tabla 5. Caso de uso 3

Identificador	*CU-04*	**Nombre**	*Detectar y reconocer señales circulares y geolocalizar*
Fuente	*CU-01, RSF-03, RSD-01, RSD-02, RSD-03*		
Actores	*Usuario*		
Objetivo	*Detectar señales de tráfico de forma circular y guardar su geolocalización*		
Precondiciones	1. *Tener la aplicación abierta* 2. *Tener la cámara habilitada* 3. *Tener el GPS y/o Internet habilitado.*		
Postcondiciones	1. *Mostrar por pantalla las señales de tráfico circulares reconocidas y guardar su identificador y geoposición en una base de datos.*		
Escenario básico	1. *Estar dentro de la aplicación* 2. *Habilitar GPS y/o Internet* 3. *Pulsar sobre la opción "Detectar círculos"*		

Tabla 6. Caso de uso 4

Identificador	*CU-05*	**Nombre**	*Detectar y reconocer señales triangulares y cuadradas*
Fuente	*CU-01, RSF-04, RSD-01*		
Actores	*Usuario*		
Objetivo	*Detectar señales de tráfico de forma triangular y cuadrada y guardar su geolocalización*		
Precondiciones	1. *Tener la aplicación abierta* 2. *Tener la cámara habilitada*		
Postcondiciones	2. *Mostrar por pantalla las señales de tráfico triangulares y cuadradas reconocidas.*		
Escenario básico	4. *Estar dentro de la aplicación* 5. *Pulsar sobre la opción "Detectar formas"*		

Tabla 7.Caso de uso 5

Identificador	*CU-06*	**Nombre**	*Detectar y reconocer señales triangulares y cuadradas y geolocalizar*
Fuente	*CU-01, RSF-04, RSD-01, RSD-02, RSD-03*		
Actores	*Usuario*		
Objetivo	*Detectar señales de tráfico de forma triangular y cuadrada y guardar su geolocalización*		
Precondiciones	1. *Tener la aplicación abierta* 2. *Tener la cámara habilitada* 3. *Tener el GPS y/o Internet habilitado.*		
Postcondiciones	1. *Mostrar por pantalla las señales de tráfico triangulares y cuadradas reconocidas y guardar su identificador y geoposición en una base de datos.*		
Escenario básico	1. *Estar dentro de la aplicación* 2. *Habilitar GPS y/o Internet* 3. *Pulsar sobre la opción "Detectar formas"*		

Tabla 8. Caso de uso 6

Identificador	*CU-07*	**Nombre**	*Visualizar base de datos*
Fuente	*CU-01, RSF-05*		
Actores	*Usuario*		
Objetivo	*Visualizar los elementos que tiene almacenados la base de datos*		
Precondiciones	1. *Tener la aplicación abierta* 2. *Tener una base de datos*		
Postcondiciones	2. *Mostrar por pantalla las señales almacenadas en la base de datos con su identificador, latitud y longitud correspondientes.*		
Escenario básico	1. *Estar dentro de la aplicación* 2. *Pulsar sobre la opción "Ver BBDD"*		

Tabla 9. Caso de uso 7

Identificador	*CU-08*	**Nombre**	*Exportar base de datos*
Fuente	*CU-01, RSF-06*		
Actores	*Usuario*		
Objetivo	*Exportar la base de datos*		
Precondiciones	1. *Tener la aplicación abierta* 2. *Tener una base de datos*		
Postcondiciones	1. *Exportar la base de datos a la tarjeta SD.*		
Escenario básico	1. *Estar dentro de la aplicación* 2. *Pulsar sobre la opción "Exportar BBDD"*		

Tabla 10. Caso de uso 8

Identificador	*CU-09*	**Nombre**	*Limpiar base de datos*
Fuente	*CU-01, RSF-07*		
Actores	*Usuario*		
Objetivo	*Exportar la base de datos*		
Precondiciones	1. *Tener la aplicación abierta* 2. *Tener una base de datos*		
Postcondiciones	1. *Eliminar registros de la base de datos*		
Escenario básico	1. *Estar dentro de la aplicación* 2. *Pulsar sobre la opción "Limpiar BBDD"*		

Tabla 11. Caso de uso 9

Identificador	*CU-10*	**Nombre**	*Calibrar cámara*
Fuente	*CU-01, RSF-08, RSD-01*		
Actores	*Usuario*		
Objetivo	*Calibrar la cámara para obtener las imágenes*		
Precondiciones	1. *Tener la aplicación abierta* 2. *Tener la cámara habilitada*		
Postcondiciones	1. *Obtención de la imagen real obtenida por la cámara*		
Escenario básico	1. *Estar dentro de la aplicación* 2. *Pulsar sobre la opción "Calibrar cámara"*		

Tabla 12. Caso de uso 9

6.4. Requisitos del sistema

En este apartado se van a definir y detallar las directrices técnicas y las directrices de gestión utilizadas, así como la catalogación de requisitos software, que servirán de base para las posteriores fases de desarrollo del proyecto.

6.4.1. Identificación de requisitos

Los requisitos de la aplicación que se enumerarán en el próximo punto tienen como propósito describir la funcionalidad y las restricciones del sistema.

Como base de la estructura genérica de un requisito software se tiene la siguiente tabla.

Identificador	*RSY-XX*	**Tipo**	*Tipo de requisito*
Nombre	*Nombre de identificador del requisito*		
Fuente	*Fuente del requisito*		
Descripción	*Descripción del requisito*		
Dependencias	*Dependencias con otros requisitos*		

Tabla 13. Representación de requisito de software genérico

En base a lo anterior tenemos lo siguiente:

- **Identificador (ID).** Es un componente del tipo RSY-XX, siendo Y el tipo de requisito software XX los números que identifican al requisito. Los valores que puede tomar Y son:
  - **F**: Funcional
  - **I**: Interfaz
  - **D**: Diseño
- **Tipo.** Define el tipo de requerimiento. Solamente puede tomar los valores.
  - **Funcional.** Describen las funcionalidades del sistema, es decir, lo que debe hacer éste.
  - **Interfaz.** Identifican a los requisitos no funcionales de interfaz.
  - **Diseño.** Identifican a los requisitos no funcionales de diseño.
- **Nombre**. Este campo contiene un nombre que define al requisito.
- **Fuente**. Origen de donde proviene el requisito.
- **Descripción**. Breve y concisa descripción del requisito.
- **Dependencias**. Identificador de los requisitos de los que depende el requisito para poder ser implementado.

6.4.2. Catalogación de los requisitos

En el presente apartado se van a catalogar todos los requisitos de usuario que se han extraído.

Debido a que puede existir un usuario con distintos perfiles, los requisitos de software funcionales (RSF) contemplarán al usuario que puede ser parte de todos ellos, es decir, un usuario que realice un uso completo de la aplicación.

Por otra parte, los *requisitos no funcionales* se han dividido en dos categorías, basándose en su descripción:

- **De interfaz**. Describen el formato con el que la aplicación se comunica con su entorno.
- **De diseño**. Describen límites o condiciones sobre cómo diseñar o implementar la aplicación.

6.4.2.1. Requisitos Software Funcionales

Identificador	*RSF-01*	**Tipo**	*Funcional*
Nombre	*Inicio de aplicación*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*Iniciar la aplicación*		
Dependencias	-		

Tabla 14. Requisito de software funcional 1

Identificador	*RSF-02*	**Tipo**	*Funcional*
Nombre	*Cierre de aplicación*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*Cerrar la aplicación*		
Dependencias	*RSF-01*		

Tabla 15. Requisito de software funcional 2

Identificador	*RSF-03*	**Tipo**	*Funcional*
Nombre	*Detectar y reconocer señales de tráfico circulares*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*Detectar e identificar las formas de tráfico de forma circular*		
Dependencias	*RSF-01, RSD-01*		

Tabla 16. Requisito de software funcional 3

Identificador	*RSF-04*	**Tipo**	*Funcional*
Nombre	*Detectar y reconocer señales de tráfico triangulares y cuadradas*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*Detectar las formas de tráfico de forma circular*		
Dependencias	*RSF-01, RSD-01*		

Tabla 17. Requisito de software funcional 4

Identificador	*RSF-05*	**Tipo**	*Funcional*
Nombre	*Visualizar base de datos*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*Visualizar los elementos que tiene almacenados la base de datos*		
Dependencias	*RSF-01*		

Tabla 18. Requisitos de software funcional 5

Identificador	*RSF-06*	**Tipo**	*Funcional*
Nombre	*Exportar base de datos*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*Exportar base de datos con señales reconocidas almacenadas*		
Dependencias	*RSF-01*		

Tabla 19. Requisito de software funcional 6

Identificador	*RSF-07*	**Tipo**	*Funcional*
Nombre	*Limpiar base de datos*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*Limpiar registros de la base de datos*		
Dependencias	*RSF-01*		

Tabla 20. Requisito de software funcional 7

Identificador	*RSF-08*	**Tipo**	*Funcional*
Nombre	*Calibrar cámara*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*Obtención de imagen de la cámara para calibración*		
Dependencias	*RSF-01, RSD-01*		

Tabla 21. Requisito de software funcional 8

6.4.2.2. Requisitos Software No Funcionales

Se procede a introducir los requisitos de software no funcionales, divididos en requisitos de portabilidad y estabilidad.

6.4.2.2.1. Requisitos Software No Funcionales de Interfaz

Identificador	*RSI-01*	**Tipo**	*Interfaz*
Nombre	*Portabilidad entre dispositivos móviles*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*La aplicación debe de funcionar en cualquier dispositivo con sistema operativo Android superior a Android 4.0.2 con cámara.*		
Dependencias	*RSF-01*		

Tabla 22. Requisitos de software no funcional de interfaz 1

Identificador	*RSI-02*	**Tipo**	*Interfaz*
Nombre	*Resolución de pantalla*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*La aplicación debe de poder ser compatible con cualquier resolución de pantalla.*		
Dependencias	*RSF-01*		

Tabla 23. Requisitos de software no funcional de interfaz 2

6.4.2.2.2. Requisitos Software No Funcionales de Diseño

Identificador	*RSD-01*	**Tipo**	*Interfaz*
Nombre	*Cámara*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*La aplicación deberá poder obtener imágenes de la cámara del dispositivo.*		
Dependencias	-		

Tabla 24. Requisitos de software no funcional de diseño 1

Identificador	*RSD-02*	**Tipo**	*Interfaz*
Nombre	*GPS*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*La aplicación deberá poder obtener datos del GPS si éste está habilitado para almacenar la geolocalización de las señales de tráfico reconocidas en la base de datos.*		
Dependencias	-		

Tabla 25. Requisito de software no funcional de diseño 2

Identificador	*RSD-03*	**Tipo**	*Interfaz*
Nombre	*Internet*		
Fuente	*Vicente Gallardo Cabrera*		
Descripción	*La aplicación deberá poder obtener datos geolocalización propocionados por Internet si éste está habilitado (datos y/o WIFI) para almacenar la geolocalización de las señales de tráfico reconocidas en la base de datos.*		
Dependencias	-		

Tabla 26. Requisito de software no funcional de diseño 3

7. DISEÑO DE LA APLICACIÓN

7.1. Introducción

En este capítulo se expondrán tanto el diseño externo como el interno de la aplicación diseñada. También se comentará, cuando sea oportuno, las vicisitudes que han hecho elegir ese tipo de diseño y no otro.

7.2. Diseño externo

En esta sección se va a detallar brevemente el diseño externo de la aplicación, considerando como diseño externo la apariencia que ésta tendrá para el usuario final.

Debido a que se ha realizado un "manual de usuario", las diferentes opciones de la aplicación y las capturas de pantalla oportunas de la misma están disponibles en el Anexo V de ésta memoria, a modo de mostrar lo mejor posible la aplicación de la manera más sencilla y breve posible así como sus funcionalidades al usuario final.

7.3. Diseño interno

Esta sección tratará del diseño interno de la aplicación, teniendo en cuenta que las alternativas iniciales de diseño interno y exponiendo los diagramas de clases, diagramas de flujo y el diseño interno de la base de datos que comprende a la aplicación.

7.3.1. Alternativas de diseño interno

Desde el primer momento se planteó la detección de formas como parte inicial de la detección de señales pudiendo así, posteriormente, extraerlas para poder reconocerlas. Esta parte inicial de diseño ha sido conservada durante todo el proceso, ya que facilita la capacidad de reconocimiento de la forma detectada y disminuye los falsos positivos.

Habiendo obtenido ya las formas y habiéndolas extraído para su análisis, el paso posterior es el del reconocimiento de la imagen extraída a una señal de tráfico conocida. Para ello se evaluó la técnica de *Detección en Cascada de Características Simples* propuesta por Paul Viola y Michael Jones en 2001 [34]. Ésta técnica conlleva un gran coste computacional a la hora del entrenamiento y un gran número de imágenes tanto positivas (contienen a la imagen objetivo) como negativas (no contienen la imagen objetivo). Para la obtención del archivo XML de cada señal a reconocer, el cual contiene la información necesaria para utilizar este tipo de

detección y poder comparar las muestras, se siguió el tutorial de Naotoshi Seo denominado *OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features)*" [36].

Después de realizar los entrenamientos y habiendo obtenido los archivos XML con el entrenamiento apropiado, se probó a ejecutar el archivo entrenado en el programa adecuado en el dispositivo Android y en ninguna de esas ocasiones funcionó debidamente. Hay que tener en cuenta que para el entrenamiento de cada señal se tuvieron previamente que tratar las imágenes negativas y positivas tal como muestra el tutorial anteriormente citado de Naotoshi Seo, y el entrenamiento de éstas, dependiendo del dispositivo, tardó entre un día y una semana, dependiendo de la cantidad de etapas del detector y la resolución de las imágenes a procesar entre otros muchos factores. Es necesario indicar que el entrenamiento de los archivos XML se realizó en una máquina con un procesador Intel Xeon E5-2430 v2 con 64GBytes de RAM y un sistema operativo Ubuntu 14.04.3 LTS, lo cual agilizó notablemente la velocidad del mismo, si no podría haber aumentado considerablemente el tiempo de entrenamiento.

Debido a la cantidad ingente de bugs que contenía el detector en cascada en OpenCV para Android, se decidió buscar una alternativa partiendo de parte del trabajo ya realizado. Teniendo ya detectadas las formas y habiéndolas extraído se podría hacer una comparación de imágenes. Tal comparación se podría llevar a cabo mediante los algoritmos descriptores de imágenes, por lo cual se decidió por utilizar ORB, debido a las pruebas que se hicieron previamente con él en Android gracias a la documentación aportada por Joseph Howse [37]. Esta línea es la que se mostrará en los puntos siguientes, y es la que se aplicará en lo sucesivo al proyecto.

7.3.2. Diagrama de clases

En esta sección se detalla el diseño interno final del sistema, mostrando la relación existente entre las distintas clases que forman la aplicación mediante el siguiente diagrama de clases.

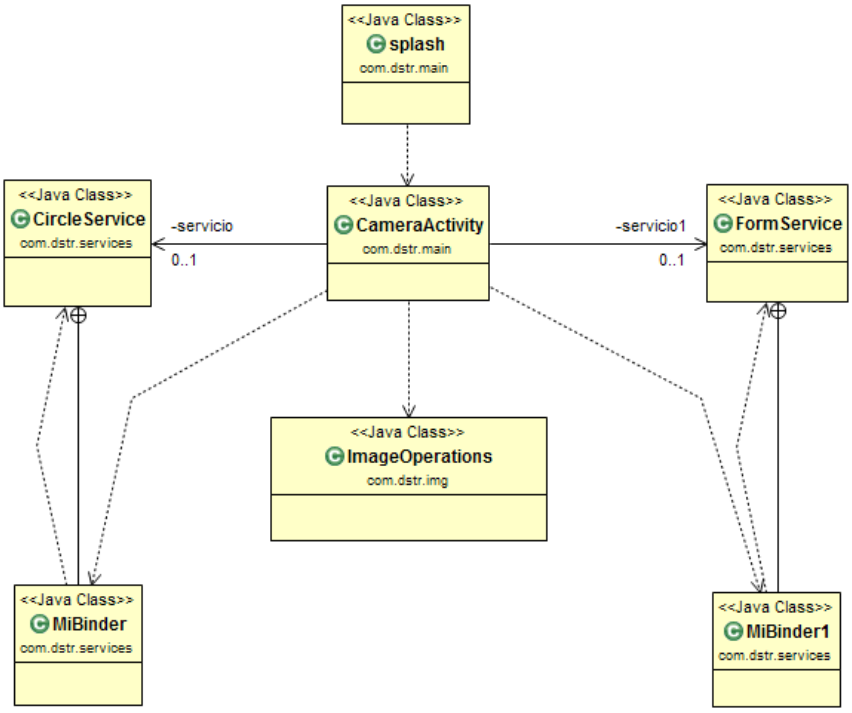


Ilustración 42. Diagrama de clases de la aplicación

Debido a la complejidad del sistema, sobretodo de cada una de sus clases, y con el fin de obtener un mejor entendimiento del diagrama anterior, a continuación se van a explicar brevemente cada una de sus partes, divididas en los paquetes que las contienen.

- **com.dstr.main:** Paquete principal de la aplicación en el que se encuentran las "*Activities*".
    - **spash.java:** Se ejecuta como la "*Activity*" inicial de la aplicación, mostrando una pantalla con el logotipo de la misma.
    - **CameraActivity.java:** Se ejecuta como la "*Activity*" principal de la aplicación. Es ejecutada después de ejecutar la clase spash.java. En ésta clase se realiza la captura de imágenes de la cámara, la gestión de la base de datos, y se llama a los "*Binder*" que enlazan con los "*Services*" para poder realizar las llamadas a sus funciones.
- **com.dstr.services:** Paquete en el que se encuentran los "*Services*" que se utilizarán para el procesamiento de las imágenes. Éstos "*Services*" son utilizados como si fueran algo similar a otra aplicación distinta, por lo que si necesitan un reinicio, no es necesario reiniciar la aplicación completa, y así se puede seguir utilizando la misma.
    - **CircleService.java:** Es el "*Service*" encargado de procesar toda la información relativa a la detección de formas circulares y reconocimiento de señales de tráfico de la misma forma.

  - **FormService.java:** Es el "*Service*" encargado de procesar toda la información relativa a la detección de formas triangulares y cuadradas y el reconocimiento de señales de tráfico de estas formas.
  - **MyBinder:** Es un "*Bind*" autogenerado que se utiliza para conectar CameraActivity.java con CircleService.java
  - **MyBinder1:** Es un "*Bind*" autogenerado que se utiliza para conectar CameraActivity.java con FormService.java
- **com.dstr.img:** Paquete que contiene las clases de identificación de las señales reconocidas.
  - **ImageOperations.java**: Esta clase es utilizada con el fin de analizar los identificadores obtenidos por los "*Services*" para identificarlos y mostrar por pantalla el identificador de cada señal.

7.3.3. Diagramas de flujo

En este apartado se procederá a exponer los diagramas de flujo pertenecientes a los procesos y subprocesos relativos a la parte de procesamiento de imagen encargada de detectar formas y reconocer las señales, las cuáles están contenidas dentro de la clase CircleService.java y FormService.java.

7.3.3.1. Detección y reconocimiento de señales circulares

El siguiente diagrama de flujo muestra el proceso por el cual se realiza la detección de formas circulares y, también, cómo posteriormente se trata la imagen obtenida de la cámara para obtener así uno o varios resultados coincidentes al reconocer las imágenes obtenidas con las imágenes de señales de tráfico precargadas por la aplicación.

Como se puede comprobar, también intervienen los procesos **máscara** y **comparación**, que se tratarán como subprocesos en este diagrama.

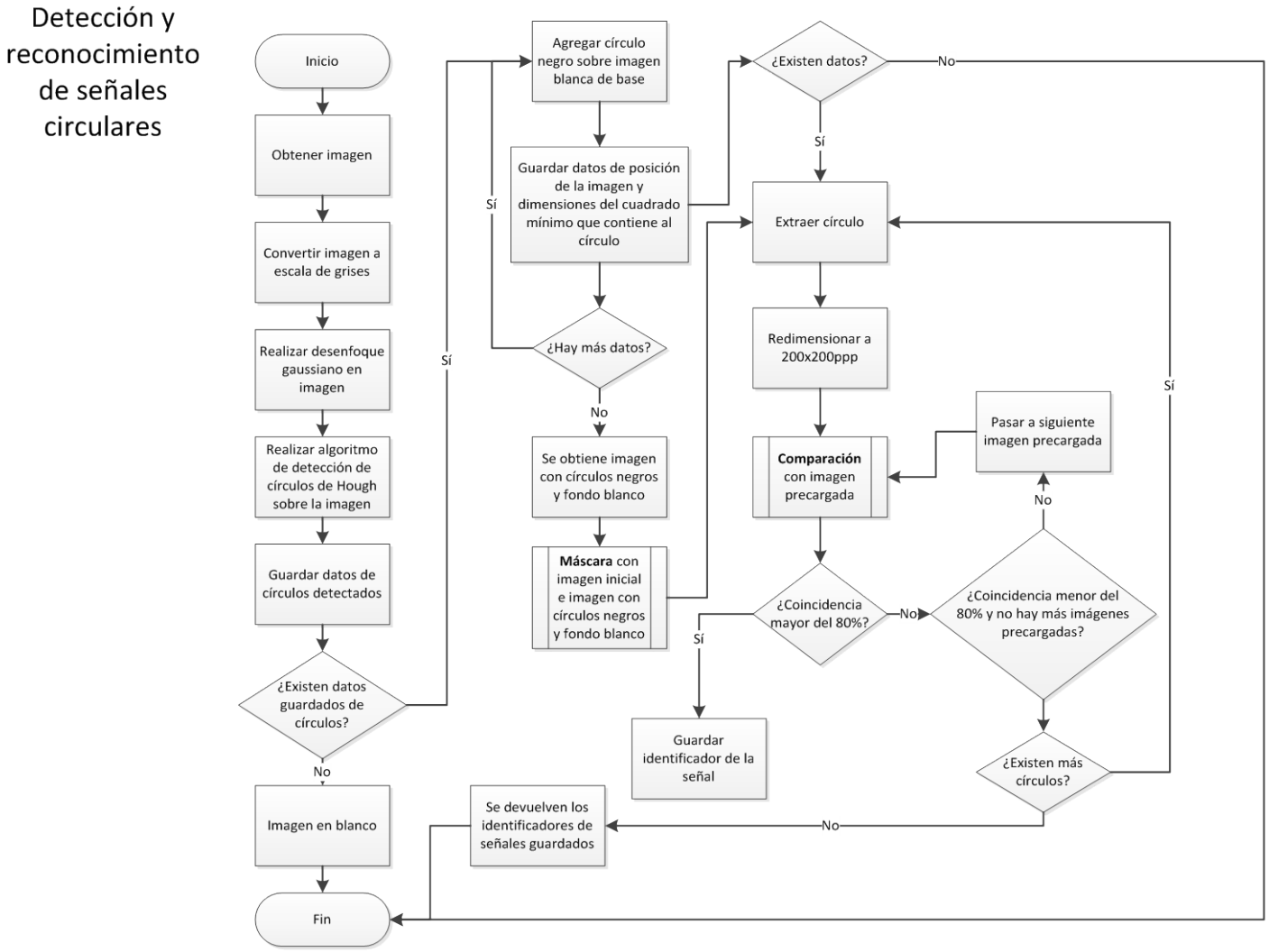


Ilustración 43. Diagrama de flujo de detección y reconocimiento de señales circulares

7.3.3.2. Detección y reconocimiento de señales triangulares y cuadradas

Este diagrama de flujo muestra el proceso por el cual se realiza la detección de formas triangulares y cuadradas (cuadriláteros en general) y, también, cómo posteriormente se trata la imagen obtenida de la cámara para obtener así uno o varios resultados con las imágenes de señales de tráfico precargadas por la aplicación.

Debido a ciertas similitudes con la detección y reconocimiento de señales circulares, se puede comprobar que también intervienen los procesos **máscara** y **comparación**, que se tratarán como subprocesos en este diagrama.

Detección y reconocimiento de triángulos y cuadrados

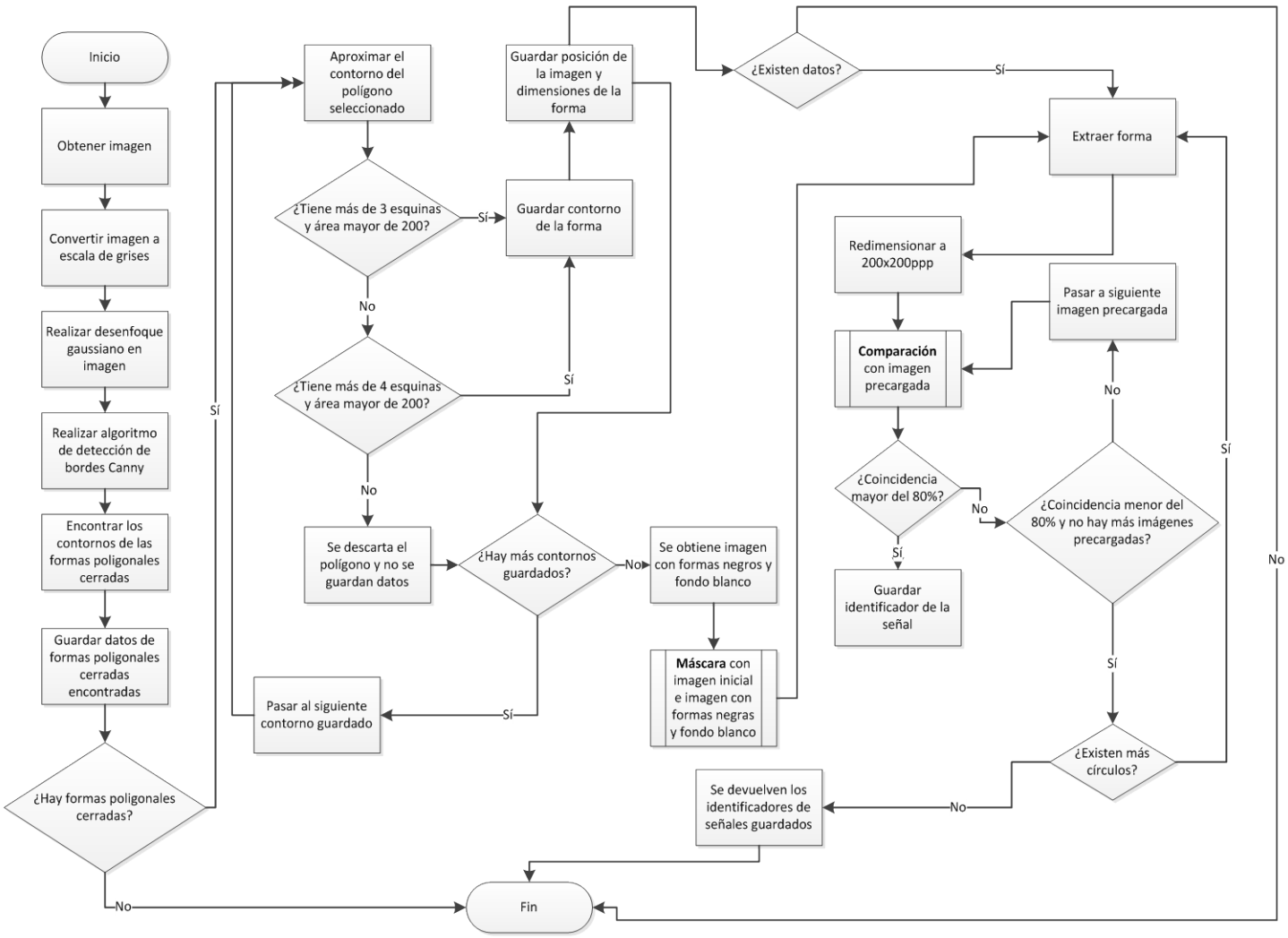


Ilustración 44. Diagrama de flujo de detección y reconocimiento de señales triangulares y cuadradas

7.3.3.3. Máscara

Este subproceso se encarga, mediante la obtención de la imagen obtenida de la cámara y de la máscara donde se encuentran seleccionadas las formas (círculos, triángulos y/o cuadrados) de mostrar éstas sobre un fondo blanco para que sean fácilmente identificables por el usuario a la hora de su visualización, y a la hora del posterior procesado para su comparación, de eliminar la mayor cantidad de información no necesaria que sea posible.

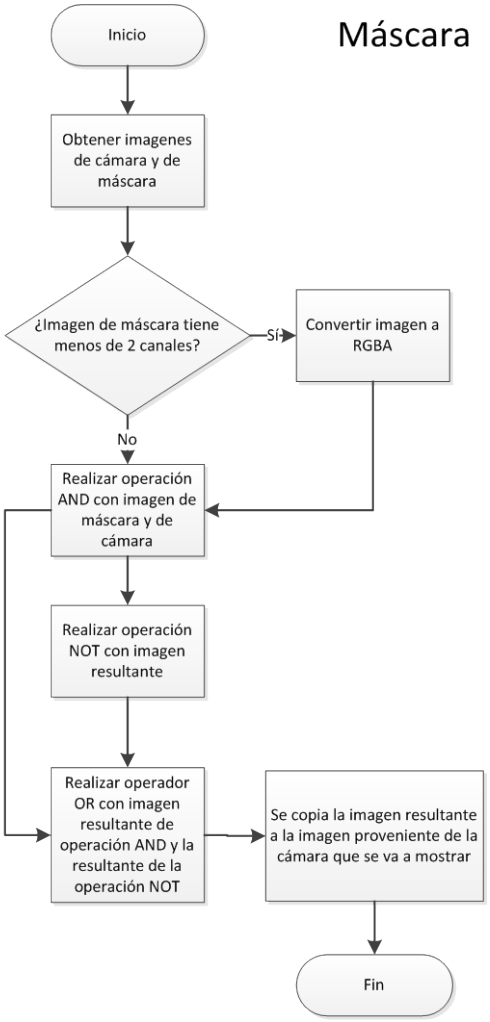


Ilustración 45. Diagrama de flujo de máscara

7.3.3.4. Comparación

Para la comparación entre imágenes se ha partido de la utilización del descriptor ORB, que se describe en la sección 5.5 de éste documento. Este comparador nos da el porcentaje de

similitud entre la imagen precargada en la aplicación y la extraída de la cámara para su comparación.

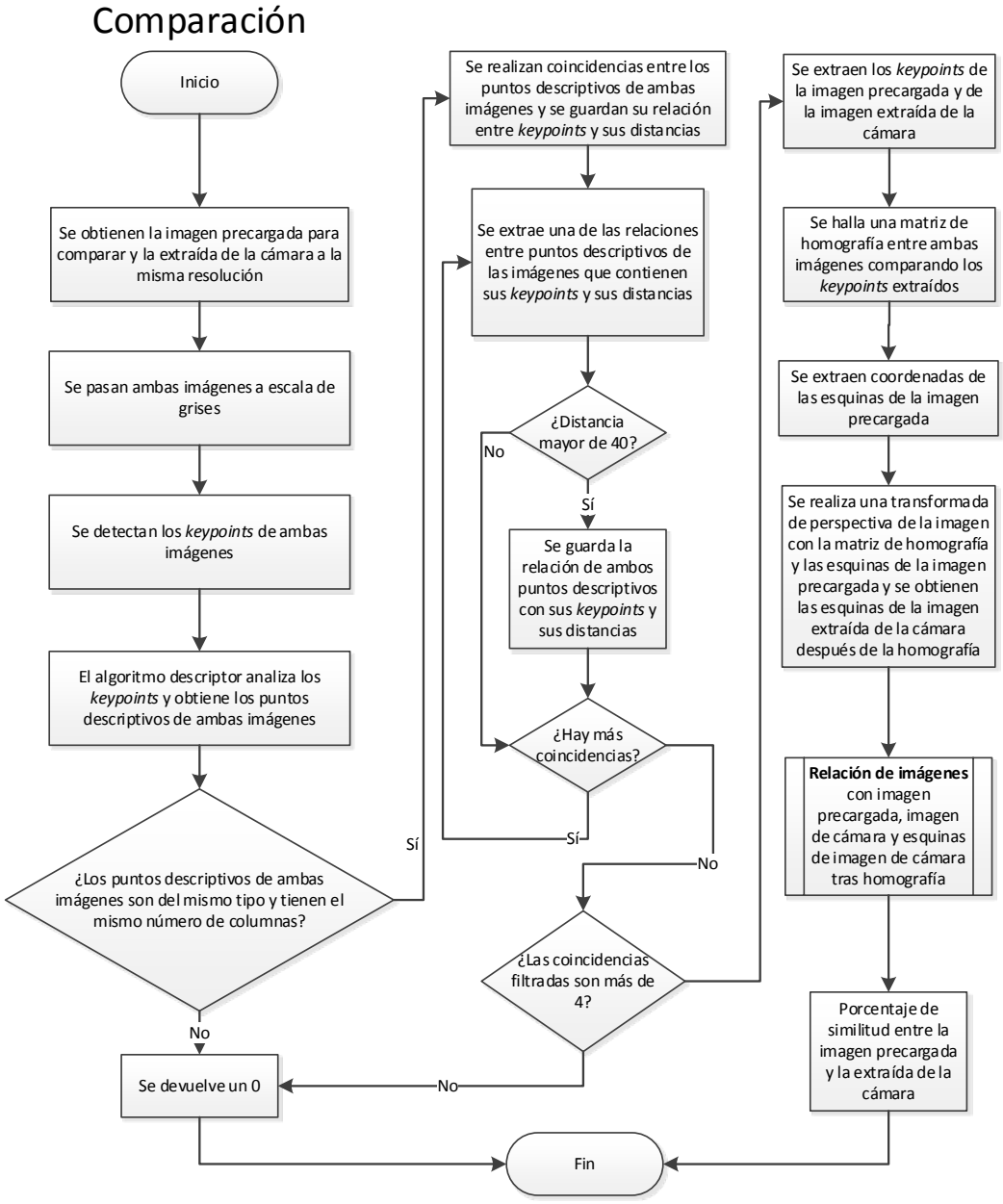


Ilustración 46. Diagrama de flujo de comparación

7.3.3.5. Relación de imágenes

Este subproceso se encarga únicamente de dar el porcentaje de relación que hay entre la imagen original a 200x200ppp y la imagen de homografía resultante. Esta homografía muestra la similitud que hay entre una imagen y otra en forma matricial, así pues, obteniendo

dos áreas de imágenes y haciendo su relación se puede obtener la diferencia proporcional entre ambas.

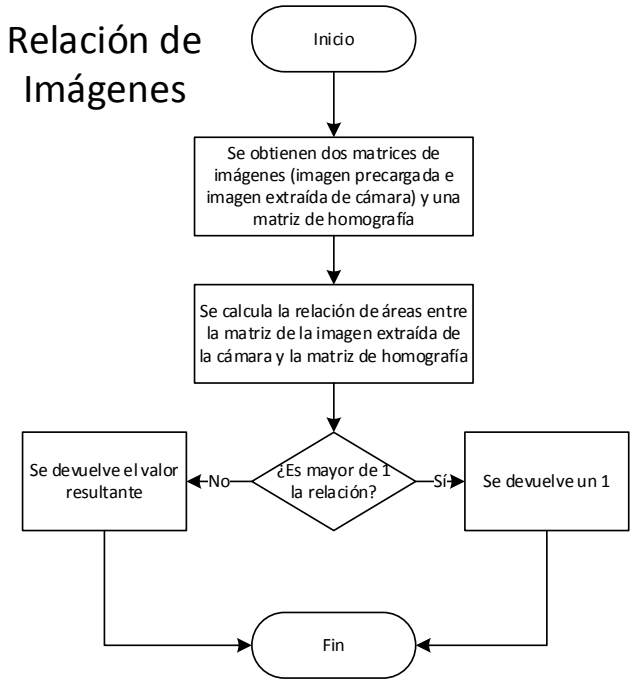


Ilustración 47. Diagrama de flujo de relación de imágenes

7.3.4. Diseño de la base de datos

Para guardar los datos de geoposicionamiento (latitud y longitud) que se obtienen después del reconocimiento de una señal, se almacenan éstos junto a uno de los identificadores de señal expuestos en el Anexo III. Se utiliza entonces una base de datos SQLite, ya que las librerías de ésta están contenidas dentro del sistema operativo Android y es de fácil lectura y escritura.

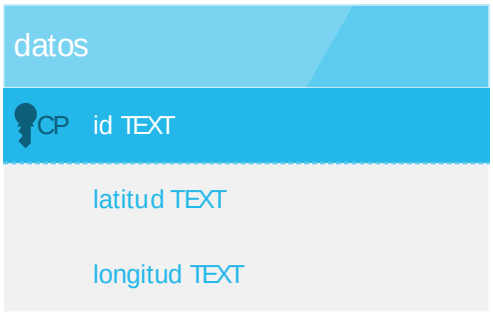


Ilustración 48. Diagrama de la base de datos

8. PRUEBAS DE LA APLICACIÓN

8.1. Introducción

En este capítulo se describirán y se expondrán las pruebas realizadas sobre el sistema centradas, básicamente, en la capacidad de detección de formas y reconocimiento de señales de tráfico.

8.2. Dispositivo utilizado

El dispositivo utilizado para la realización de las pruebas es un Smartphone Sony Xperia Z1, con una resolución seleccionada de captura de imagen de 720x480ppp. En el Anexo IV adjunto se contemplan las demás especificaciones técnicas del dispositivo.

8.3. Detección de formas

Primeramente, se analizará la capacidad de detección de las formas a tratar por la aplicación, a saber: círculos, triángulos y cuadrados (cuadriláteros en general).

Para calcular el porcentaje de obtención de las formas de la siguiente tabla se ha realizado la comparación sobre 100 muestras de imágenes capturadas por la aplicación con unas condiciones de iluminación adecuadas, sobre formas reconocibles no mayores de 8 cm y no menores de 4 cm, a una distancia menor a 0,5 metros sobre una pantalla TFT.

Forma	Aciertos (detectados)	Fallos (no detectados)	Falsos positivos
Círculo	82%	28%	8%
Triángulo	73%	27%	2%
Cuadrado	68%	32%	5%

Tabla. Porcentaje de aciertos, fallos y falsos positivos en la detección

Estos datos obtenidos varían sensiblemente a la hora de detectar las formas en entornos urbanos e interurbanos, pudiendo disminuir la precisión aproximadamente entre un 10 y un 30 por ciento dependiendo de los objetos situados en la parte superior de la imagen, de lo deteriorada que esté ésta (siempre que sea reconocible) y la iluminación a contra o a favor de la figura.

También se ha podido observar que la detección de formas en entornos urbanos e interurbanos puede oscilar su primera detección entre los 10 y 20 metros de distancia entre la

señal a reconocer y el dispositivo detector (móvil), dependiendo de lo expuesto anteriormente y, también, de las dimensiones que tenga la señal, pues estas no tienen una medida estándar.

Es de destacar que las formas triangulares y cuadradas también se pueden detectar de manera lateral, cosa que no sucede con las formas circulares, que se detectan siempre de manera frontal.

8.4. Reconocimiento de señales de tráfico

Para realizar las pruebas de reconocimiento se ha tenido en cuenta primeramente el haber detectado la forma. Las condiciones de captura de imagen son las mismas que en el apartado anterior, a excepción de que las imágenes de prueba son las mismas imágenes de señales de tráfico que están precargadas en la aplicación. Sabiendo esto, y debido a que el reconocimiento de señales circulares y el de señales triangulares y cuadradas está diferenciado en la aplicación se procederá a completar dos tablas diferentes.

Con fin de concretar más, se procederá a pasar 1 muestra durante 100 fotogramas de una de las imágenes precargadas para ver su número de aciertos y fallos. Posteriormente se pasarán 100 imágenes aleatorias distintas, entre las precargadas en la aplicación, para ver cuántos falsos positivos hay con respecto al identificador de la imagen.

Referencia de la señal	Aciertos (detectados)	Fallos (no detectados)	Falsos positivos
R-400c	56%	44%	8%
R-400d	61%	39%	6%
R-402	58%	42%	12%
R-407a	82%	18%	2%

Tabla 27. Porcentaje de aciertos, fallos y falsos positivos en el reconocimiento de señales de tráfico circulares

Referencia de la señal	Aciertos (detectados)	Fallos (no detectados)	Falsos positivos
P-18	81%	19%	4%
P-21	79%	21%	9%
P-22	89%	11%	6%
P-50	64%	36%	4%
S-11	79%	21%	15%
S-11a	34%	66%	2%
S-17	11%	89%	1%

Tabla 28. Porcentaje de aciertos, fallos y falsos positivos en el reconocimiento o de señales de tráfico triangulares y cuadradas

Hay que destacar también que en las pruebas realizadas se pueden obtener resultados para diferentes señales que se han detectado a la vez, mostrando así diferentes datos por pantalla.

También se ha medido la velocidad de procesamiento de las imágenes obtenidas por la cámara, obteniendo así una cantidad de entre 9,04 y 1,08 imágenes por segundo procesadas, dependiendo del entorno en el que se encuentren las imágenes a capturar y la cantidad de ellas que haya.

Es importante tener en cuenta que las señales circulares solo se reconocen frontalmente, mientras que las señales triangulares y cuadradas tienen la capacidad de reconocerse también si están ladeadas.

Debido a la característica especial que tiene ORB para encontrar pares de muestras coincidentes en imágenes rotadas es posible encontrar comparaciones en señales que estén torcidas, giradas o ladeadas. Esto es también una contrapartida, debido a que algunas señales pueden detectarse dando falsos positivos ya que son una variante de la misma señal original pero rotada con una cierta inclinación.

9. CONCLUSIONES Y LÍNEAS FUTURAS

9.1. Introducción

A lo largo de este capítulo se expondrán las conclusiones adquiridas durante el desarrollo del proyecto, tanto a nivel del entorno que enmarca al TFG como de las propias experiencias del autor. Finalmente, en el apartado de líneas futuras se contemplan todas aquellas funcionalidades deseables o planteadas para añadir valor y funcionalidad a la aplicación desarrollada.

9.2. Conclusiones

En el primer capítulo del presente documento se citaron los principales objetivos de los que constaba el TFG desarrollado y es en el punto actual donde se realiza un análisis en perspectiva del trabajo realizado, describiendo todos aquellos logros conseguidos y las conclusiones que este aporta.

El principal objetivo a conseguir era desarrollar una aplicación que fuera capaz de detectar señales de tráfico verticales con el afán de ayudar a la conducción, ya sea mejorando la conducción, reduciendo el consumo, disminuyendo el número de accidentes en las vías de circulación y/o disminuyendo el estrés del conductor, siendo posible su instalación en cualquier dispositivo con sistema operativo Android (ya sean smartphones o tablets). Este objetivo ha sido el pilar durante todo el proyecto, teniendo así el desarrollo íntegro de una aplicación sencilla y mantenible para poder ser utilizado en cualquier dispositivo Android con una cámara trasera de una resolución más o menos aceptable.

Asimismo, hay que tener presente que este trabajo no hubiera sido posible, al menos de la envergadura que puede llegar a alcanzar a corto o medio plazo, de no ser por la dedicación y el interés en facilitar documentación y ayuda de mi tutor de proyecto, D. Víctor Corcoba Magaña, el cual me ha ayudado en todo lo que ha estado en su mano. También agradecer la ayuda al Departamento de Telemática de la UC3M, por concederme temporalmente acceso a uno de sus servidores para poder realizar algunas de las pruebas del proyecto. De especial relevancia es también el agradecimiento a la comunidad de OpenCV por facilitar unas librerías para Android, lo cual es de especial relevancia para el desarrollo de este trabajo.

Respecto a las dificultades que se han experimentado durante el proceso de desarrollo del proyecto, la primera barrera a superar ha sido entender las tecnologías actuales aplicadas (Android y OpenCV) y saber si éstas serían las correctas para el desarrollo del proyecto, aunque

no menos relevante ha sido el tiempo para aprenderlas y la dificultad que se ha tenido a la hora de utilizarlas.

El proyecto se enmarca en Android con una librería de OpenCV para este sistema operativo. El desconocimiento de ambas ha llevado a un intensivo estudio de las mismas y a formarme sobre procesamiento de imagen digital y programación en Android.

También hay que destacar las limitaciones de hardware, tanto del PC donde se ha desarrollado el proyecto en su mayor parte (a excepción del entrenamiento del detector en cascada), como del dispositivo móvil utilizado para el mismo los cuales necesitaban grandes tasas de rendimiento que no tenían.

Por último, la valoración de la experiencia propia durante el transcurso de desarrollo del proyecto es muy positiva. A pesar de los contratiempos que han surgido, estos también han servido para aprender. Ha sido un proyecto muy completo respecto a conocimientos adquiridos y asentados y de retos por superar. Además, en un futuro se pretende seguir con ésta línea de investigación para futuras mejoras.

9.3. Líneas futuras

Una vez manifestadas las conclusiones sobre el proceso del desarrollo del TFG, en este apartado se van proponer líneas futuras sobre las que trabajar para dotar a la aplicación desarrollada de mayor funcionalidad.

Se ha planteado que cuando el dispositivo tenga conectado el GPS y/o internet y detecte la geolocalización el dispositivo muestre por pantalla las señales de tráfico almacenadas en la base de datos de la aplicación comparando los datos de obtenidos por el GPS (latitud y longitud), basado en un factor de corrección que tenga en cuenta la cercanía y la dirección de desplazamiento, mostrando las más cercanas.

Como complemento a lo anterior se desearía implementar la capacidad de importar una base de datos distinta a la de registro de datos. Ésta tendrá la información corroborada de las señales existentes en las vías y carecerá de información redundante para una mejor consulta.

También se ha planteado darle un mejor aspecto visual a la aplicación, de modo que sea más atractiva y más intuitiva para el usuario final.

Es de especial relevancia mejorar la eficacia de la aplicación, por lo que se estudiarán nuevos algoritmos más eficaces y eficientes en cuanto a capacidad de detección y velocidad de procesamiento se refiere.

También se estudiará la capacidad de ampliar la capacidad de detección implementando dispositivos de aumento óptico a larga distancia para detectar y reconocer con mayor antelación las señales que se presenten en las vías urbanas e interurbanas.

Como complemento para fomentar la conducción responsable, se evaluará implementar medidas de gamificación, como se puede ver en aplicaciones como Waze o Swarm (antiguo Foursquare) para compartir datos con otros conductores y poder así comparar a los mismos, pudiendo darles una puntuación en base a su tipo de conducción y a la cantidad de señales de tráfico nuevas que aporten al archivo común, entre otros.

9. CONCLUSIONS AND FUTURE WORK

9.1. Introduction

Throughout this chapter, conclusions acquired during the project will be presented, in terms of both project environment and author's own experiences. Finally, in the "future work" paragraph all the desirable or planned features that add value and functionality to the application will be discussed.

9.2. Conclusions

During the first chapter of this memory we could see the main objectives that the project pursued. Once developed, this topic will analyze in perspective all the work done, describing the achievements and conclusions extracted.

The main goal to achieve was to develop an application capable of detecting vertical road signs in an effort to help the driver, either by improving driving, reducing fuel consumption, reducing the number of accidents on the roads or releasing driver stress, and compatible with any device with Android operating system (smartphones or tablets). Obtaining a full simple and maintainable application which can be used on any Android device with a rear camera of an acceptable resolution has been one of the most important objectives throughout the project.

I would also like to thank my tutor project, Mr. Victor Corcoba Magaña, for his dedication and interest in facilitating documentation and support in everything that has been in his hand. I would also like to acknowledge the assistance of the Department of Telematics of the UC3M, for temporarily grant access to one of their servers to perform some of the tests of the project. A special thanks to the community by providing the Android's library OpenCV, essential for the development of this work. Without their help and dedication, this work would not have been possible, at least on the scale it could reach in the short or medium term.

Regarding difficulties, the main obstacle to tackle was to understand current technologies applied (Android and OpenCV) and whether they would be appropriate or not for the project. Although at least as important was the amount of time used to learn that technologies and the difficulty to master them.

The project uses OpenCV library for Android operating system. The lack of knowledge in both technologies has led to an intensive study and a deep learning about digital image processing and programming on Android.

It is also noticeable the limitations of hardware, both PC and mobile device where the project was developed for most of the time (except for cascade detector training) lacked from great performance rates.

As a final conclusion, the assessment of the experience over the course of the project was very positive. The setbacks arisen have been also a good lesson to learn. It was a very comprehensive project, challenging and full of knowledge to acquired and retain. In addition, this line of research can be continued for future improvements.

9.3. Future work

Once conclusions are shown, this section will propose future work in order to provide enhanced functionality to the application.

One possible line is, whenever the device is connected to GPS and/or internet and detects geolocation, to compare data obtained of GPS with traffic signals stored in the database and then display signals matched on screen, based on a correction factor, closeness and direction of travel.

To complement the above feature, it would be desirable for the application to be able to import a signal database other than the log data. For a better query, this information will be accurate and will lack of redundant information.

Another proposal is to give a better visual appearance to the application, so that it is more attractive and more intuitive for end user.

In order to improve the application in terms of detection capability and processing speed, new more effective and efficient algorithms will be studied.

The ability of implementing long distance detection optical magnifying devices for a much earlier signal detection and recognition in urban and inner city roads will also be studied.

As a complement to encourage responsible driving, gamification measures will be evaluated, as seen in other applications such as Waze or Swarm (formerly Foursquare) to share data with other drivers so they can compare them and get a score based on driving style and amount of new traffic signals contributed to the common store.

Bibliografía

[1] M. Batty, K. Axhausen, G. Fosca, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis y Y. Portugali, «Smart Cities of the Future», *UCL WORKING PAPERS SERIES,* nº 188, 2012.

[2] Google, «Google Self-Driving Car Project», [En línea]. Available: http://www.google.com/selfdrivingcar/. [Último acceso: 17 agosto 2015].

[3] Á. Arcos García, M. Soilán, B. Rivero, J. A. Álvarez-García, J. Y. Fernández, J. A. Ortega y P. Arias-Sánchez, «Sistema de reconocimiento de señales de tráfico para una Smart City».

[4] Bosch, «myDriveAssist», [En línea]. Available: http://iphone.bosch.com/mydriveassist/index.html. [Último acceso: 17 agosto 2015].

[5] R. Hervás, «DriveAid: Aplicación Android de detección y reconocimiento de señales de tráfico», [En línea]. Available: http://mami.uclm.es/rhervas/index.php/docencia/trabajos-y-proyectos-fin-de-carrera/98-pfc-emozos. [Último acceso: 1 septiembre 2015].

[6] Volvo, «Volvo Cars Support», [En línea]. Available: http://support.volvocars.com/. [Último acceso: 30 agosto 2015].

[7] BMW, «BMW Website», [En línea]. Available: http://www.bmw.com. [Último acceso: 2015 agosto 28].

[8] IDC Corporate USA, [En línea]. Available: http://www.idc.com. [Último acceso: 10 septiembre 2015].

[9] NVIDIA, «NVIDIA Tegra Android Development Pack», [En línea]. Available: https://developer.nvidia.com/tegra-android-development-pack. [Último acceso: 20 marzo 2015].

[10] Wikimedia Commons, «Wikimedia», [En línea]. Available: https://commons.wikimedia.org. [Último acceso: 2015 septiembre 2].

[11] Google, «Google Play», [En línea]. Available: https://play.google.com/. [Último acceso: 30 agosto 2015].

[12] YouTube, «YouTube», [En línea]. Available: https://www.youtube.com/. [Último acceso: 30 agosto 2015].

[13] automotiveIT International, «automotiveIT International», [En línea]. Available: http://www.automotiveit.com. [Último acceso: 1 septiembre 2015].

[14] Google, «Android», [En línea]. Available: https://www.android.com/intl/es_es/. [Último acceso: 1 septiembre 2015].

[15] Kantar, «Kantar Worldpanel», [En línea]. Available: http://www.kantarworldpanel.com/global/smartphone-os-market-share/. [Último acceso:

19 agosto 2015].

[16] The Eclipse Foundation, «Eclipse», [En línea]. Available: http://www.eclipse.org/home/index.php. [Último acceso: 30 marzo 2015].

[17] OpenCV, «OpenCV.org», [En línea]. Available: http://opencv.org/. [Último acceso: 1 septiembre 2015].

[18] G. Bradski y A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, Safari Books Online, 2008.

[19] SQLite, «SQLite,» [En línea]. Available: https://www.sqlite.org/about.html. [Último acceso: 18 agosto 2015].

[20] A. B. Adams, High-dimensional Gaussian Filtering for Computational Photography, Standford University, 2011.

[21] J. Canny, «A Computational Approach To Edge Detection», *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 8, nº 6, pp. 679-698, 1986.

[22] R. C. Gonzalez y R. E. Woods, Digital Image Processing, Third Edition, Pearson Education, 2010.

[23] OpenCV, «OpenCV: Canny Edge Detector», [En línea]. Available: http://docs.opencv.org/master/da/d22/tutorial_py_canny.html. [Último acceso: 1 septiembre 2015].

[24] P. Hough, «Method and means for recognizing complex patterns». Estados Unidos Patente 3, 069 654, 18 diciembre 1962.

[25] R. O. Duda y H. P. E., «Use of the Hough Transformation to Detect Lines and Curves in Pictures», *Communications of the ACM,* vol. 15, pp. 11-15, 1972.

[26] D. Ballard, «Generalizing the Hough Transform to Detect Arbitrary Shapes», *Pattern Recognition,* vol. 13, nº 2, pp. 111-122, 1981.

[27] O. Espinar Mosquera, «mundo-R», [En línea]. Available: http://www.xente.mundo-r.com/Oscar/index.html. [Último acceso: 2015 agosto 28].

[28] E. Rublee, V. Rabaud, K. Konolige y G. Bradski, *ORB: an efficient alternative to SIFT or SURF,* Menlo Park, California: Willow Garage, 2011.

[29] E. Rosten y T. Drummond, «Machine learning for high-speed corner detection», de *European Conference on Computer Vision*, 2006.

[30] M. Calonder, V. Lepetit, C. Strecha y P. Fua, «Brief: Binary robust independent elementary feature», de *European Conference on Computer Vision*, 2010.

[31] D. G. Lowe., «Distinctive image features from scale-invariant keypoints», *International*

Journal of Computer Vision, vol. 2, nº 60, pp. 91-110, 2004.

[32] H. Bay, T. Tuytelaars y L. V. Gool, «Surf: Speeded up robust features», de *European Conference on Computer Vision*, 2006.

[33] G. Levi, «Gil's Computer vision blog», [En línea]. Available: https://gilscvblog.wordpress.com. [Último acceso: 2015 agosto 28].

[34] P. Viola y M. Jones, «Rapid Object Detection using a Boosted Cascade of Simple», de *ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION*, 2001.

[35] Institute of Electrical and Electronics Engineers, Inc, «IEEE Std 830-1998, Recommended Practice for Software Requirements Specifications», 1998.

[36] N. Seo, «Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features)», [En línea]. Available: http://note.sonots.com/SciSoftware/haartraining.html. [Último acceso: 22 mayo 2015].

[37] J. Howse, Android Application Programming with OpenCV, Packt Publishing, 2013.

[38] H. Dayami y E. Fleyeh, «Eigen-based traffic sign recognition», *IET Intelligent Transport Systems,* vol. 5, nº 3, p. 190–196, 2011.

ANEXO I. ACRÓNIMOS Y ABREVIATURAS

En este anexo se procederá a confeccionar una tabla de acrónimos y definiciones para comprender de la mejor forma posible el actual documento.

1. Acrónimos

A continuación se listan los acrónimos que se emplean a lo largo del documento:

Acrónimo	Definición
ADT	Android Development Tools
API	Application Programming Interface
BMP	Bitmap
BRIEF	Binary Robust Independent Elementary Features
CDT	C/C++ Development Tooling
CU	Caso de Uso
dpi	Dot Per Inch
DSTr	Detector de Señales de Tráfico
DVM	Dalvik Virtual Machine
ECJ	Eclipse Compiler for Java
FAST	Features from Accelerated Segment Test
GPS	Global Positioning System
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
JDK	Java Development Kit
JDT	Java Development Toolkit
JPG-JPEG	Joint Photographic Experts Group
JVM	Java Virtual Machine
NDK	Native Development Kit
OpenCV	Open Source Computer Vision
ORB	Oriented FAST and Rotated BRIEF
OS	Operating System
PNG	Portable Network Graphics
ppp	Píxeles Por Pulgada
rBRIEF	Rotated BRIEF
RGB	Red Green Blue
RGBA	Red Green Blue Alpha
RSD	Requisito de Software No Funcional de Diseño
RSF	Requisito de Software Funcional
RSI	Road Sign Information
RSI	Requisito de Software No Funcional de Interfaz
RST	Reconocimiento de Señales de Tráfico
SDK	Software Development Kit
SIFT	Scale Invariant Feature Transform
SO	Sistema Operativo
SQL	Structured Query Language
SURF	Speeded-Up Robust Features

TADP	Tegra Android Development Pack
TFG	Trabajo de Fin de Grado
USB	Universal Serial Bus
XML	eXtensible Markup Language

Tabla 29. Acrónimos

2. Definiciones

En este apartado se van a incluir las definiciones de los términos que se emplean en el documento para una mayor comprensión del mismo:

- **Application Programming Interface:** es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
- **eXtensible Markup Language (XML):** es un lenguaje de marcas desarrollado por el World Wide Web Consortium.
- **Histéresis:** fenómeno por el que el estado de un material depende de su historia previa.
- **Histograma:** representación gráfica de la distribución de los distintos tonos de una imagen.
- **Integrated Development Environment:** es un entorno de programación que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.
- **iOS:** sistema operativo desarrollado por la empresa Apple para dispositivos iPhone, iPod Touch e iPad.
- **keypoint:** en un punto clave de una imagen, es decir, un punto característico de la misma.
- **Ubuntu:** Distribución del sistema operativo GNU/Linux, basado en Debian. Linux se presenta como la gran alternativa a los sistemas operativos propietarios, y está altamente extendido en muchas de sus distribuciones sobre todo en servidores. Esta distribución fue creada por Mark Shuttleworth, con el fin de llevar Linux al mayor número de usuarios posibles, simplificando todos los procesos de instalación y manejo.
- **Unix:** es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969, por un grupo de empleados de los laboratorios Bell de AT&T.
- **Windows:** sistema operativo cuya marca propietaria es Microsoft.

ANEXO II. GESTIÓN DEL PROYECTO

1. Planificación del proyecto

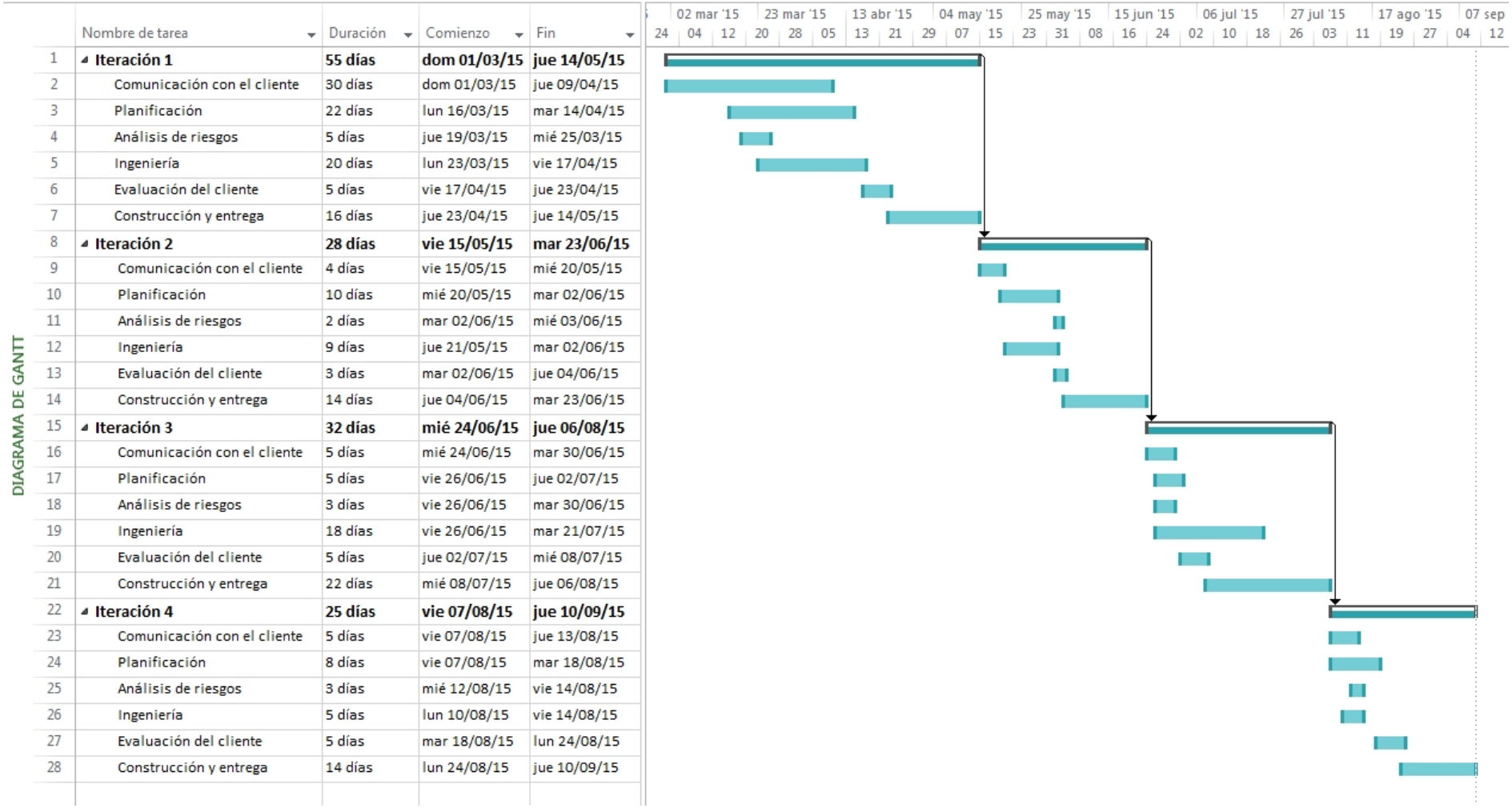
En este apartado se detallará el diagrama de Gantt basado en las fases que incluye este proyecto, atendiendo a la metodología de desarrollo aplicada. Para cada una de las subtareas que incluyen las fases, se mostrará la duración y el periodo de tiempo en el que han sido elaboradas.

Se han previsto cuatro iteraciones para el desarrollo íntegro del sistema:

- **Iteración 1.** Se ocupa de la definición del alcance del proyecto, así como de los objetivos, tecnologías a usar y aprendizaje de las mismas, es decir, fases de análisis y diseño inicial.
- **Iteración 2.** Enfocada mayormente a la especificación del diseño, implementación de funcionalidades y pruebas del sistema.
- **Iteración 3.** Está dirigida a realizar pruebas sobre el sistema ya existente y a la búsqueda de nuevos algoritmos para el desarrollo de la aplicación.
- **Iteración 4.** Dedicada a la implementación final del producto, mejoras, corrección de errores, pruebas y documentación formal.

Respecto a cada una de las fases en las que se divide cada iteración, se encuentran las siguientes:

- **Comunicación con el cliente.** Definición de los objetivos de cada una de las iteraciones del proyecto.
- **Planificación.** Organización de las tareas relativas a cada iteración.
- **Análisis de riesgos.** Evaluación de las alternativas en la implementación y sus posibles riesgos asociados.
- **Ingeniería.** Fase de aprendizaje de nuevas tecnologías, definición de requisitos y de análisis de componentes y clases de la aplicación.
- **Evaluación del cliente.** Aceptación y evaluación, por parte del cliente (el propio desarrollador), de la documentación proporcionada en la iteración.
- **Construcción y entrega.** Implementación y documentación formal de la funcionalidad especificada en fases anteriores.



	Nombre de tarea	Duración	Comienzo	Fin
1	**Iteración 1**	**55 días**	**dom 01/03/15**	**jue 14/05/15**
2	Comunicación con el cliente	30 días	dom 01/03/15	jue 09/04/15
3	Planificación	22 días	lun 16/03/15	mar 14/04/15
4	Análisis de riesgos	5 días	jue 19/03/15	mié 25/03/15
5	Ingeniería	20 días	lun 23/03/15	vie 17/04/15
6	Evaluación del cliente	5 días	vie 17/04/15	jue 23/04/15
7	Construcción y entrega	16 días	jue 23/04/15	jue 14/05/15
8	**Iteración 2**	**28 días**	**vie 15/05/15**	**mar 23/06/15**
9	Comunicación con el cliente	4 días	vie 15/05/15	mié 20/05/15
10	Planificación	10 días	mié 20/05/15	mar 02/06/15
11	Análisis de riesgos	2 días	mar 02/06/15	mié 03/06/15
12	Ingeniería	9 días	jue 21/05/15	mar 02/06/15
13	Evaluación del cliente	3 días	mar 02/06/15	jue 04/06/15
14	Construcción y entrega	14 días	jue 04/06/15	mar 23/06/15
15	**Iteración 3**	**32 días**	**mié 24/06/15**	**jue 06/08/15**
16	Comunicación con el cliente	5 días	mié 24/06/15	mar 30/06/15
17	Planificación	5 días	vie 26/06/15	jue 02/07/15
18	Análisis de riesgos	3 días	vie 26/06/15	mar 30/06/15
19	Ingeniería	18 días	vie 26/06/15	mar 21/07/15
20	Evaluación del cliente	5 días	jue 02/07/15	mié 08/07/15
21	Construcción y entrega	22 días	mié 08/07/15	jue 06/08/15
22	**Iteración 4**	**25 días**	**vie 07/08/15**	**jue 10/09/15**
23	Comunicación con el cliente	5 días	vie 07/08/15	jue 13/08/15
24	Planificación	8 días	vie 07/08/15	mar 18/08/15
25	Análisis de riesgos	3 días	mié 12/08/15	vie 14/08/15
26	Ingeniería	5 días	lun 10/08/15	vie 14/08/15
27	Evaluación del cliente	5 días	mar 18/08/15	lun 24/08/15
28	Construcción y entrega	14 días	lun 24/08/15	jue 10/09/15

Ilustración 49. Diagrama de Gantt con planificación de proyecto

2. Presupuesto del proyecto

En el presente apartado de éste anexo se va a detallar el impacto económico que ha supuesto este Trabajo de Fin de Grado. Para ello, se va a proceder a realizar un análisis de los costes de personal, de equipos y de licencias de software.

En primer lugar, se incluye una descripción del trabajo realizado, donde se plasmará el nombre del proyecto, el autor y la duración estimada del mismo, así como la tasa de costes indirectos aplicada al presupuesto total.

Nombre del trabajo	Extracción de información de las carreteras para la reducción del consumo de combustible y mejora en la seguridad
Autor	Vicente Gallardo Cabrera
Duración	6,34 meses
Tasa de costes indirectos	20%

Tabla 30. Descripción del proyecto realizado

Seguidamente se abordará el cálculo de los costes de personal del proyecto, teniendo en cuenta las siguientes consideraciones:

- Duración planificada: 6,34 meses (142 días laborables).
- Horas al día (media): 3,5 horas/día.
- Dedicación (1 hombre/mes): 78 horas.
- Coste de 1 hombre jornada completa/mes laboral: 2.230,41€.
- El coste de personal se calcula atendiendo a la siguiente fórmula:

$$Coste\ de\ personal = \frac{Duración\ x\ Horas\ al\ día}{Dedicación}\ x\ Coste\ (hombre/mes\ laboral)$$

El coste de personal del proyecto es de CATORCE MIL DOSCIENTOS ONCE EUROS CON SETENTA Y DOS CÉNTIMOS DE EUROS.

Respecto a los gastos de equipos, a continuación se muestran los gastos de equipos concernientes a la realización del proyecto:

Descripción	Coste sin IVA (€)	Uso dedicado al proyecto	Dedicación (meses)	Periodo de depreciación (meses)	Coste imputable (€)
PC Intel Dual Core + Pantalla + Ratón + Teclado	882,28€	68%	90	60	0,00
Sony Xperia Z1	329,83	61%	9	24	75,45
Servidor Intel Xeon E5-2430 v2	3580,00	24%	3	60	42,96

Tabla 31. Coste de material del proyecto

Para el cálculo del coste imputable de los gastos de equipos, se ha aplicado la siguiente fórmula:

$$Coste\ imputable = \frac{Dedicación}{Periodo\ de\ depreciación}\ x\ Coste\ \text{sin}\ IVA\ x\ \%\ Uso\ dedicado$$

El resultado total del coste imputable de gastos de material es de CIENTO DIECIOCHO EUROS CON CUARENTA Y UN CÉNTIMOS DE EURO.

En la siguiente tabla se van a mostrar todos los gastos que ha habido en licencias de software para el desarrollo del proyecto:

SOFTWARE	Coste sin IVA (€)
Android 4.4.2	0,00
Microsoft Windows 7 Ultimate	289,28
Ubuntu 14.04 LTS	0,00
Microsoft Office 2013	192,81
NVIDIA TADP	0,00
SQLite	0,00
Photoshop CS6	563,84

Tabla 32. Coste de licencias de software utilizado en el proyecto

El precio total en gastos de licencias es de MIL CUARENTA Y CINCO EUROS CON NOVENTA Y TRES CÉNTIMOS DE EURO

Por último, se va a mostrar un resumen del presupuesto con la suma de todos los costes asociados al proyecto. Destacar que el IVA aplicado al presupuesto del proyecto es del 21%.

Concepto	Coste sin IVA (€)	IVA (€)	Coste con IVA (€)
Personal	14.211,72	2.984,46	17.196,18
Material	118,41	24,87	143,28
Licencias	1045,93	219,65	1.265,58
Costes indirectos	3.075,22	645,80	3.721,02
Total	18.451,28	3.874,77	22.326,05

Tabla 33. Resumen del presupuesto del proyecto

El coste total del proyecto, contabilizando gastos de personal, equipos, material y licencias es de 22.326,05€ (VEINTIDOS MIL TRESCIENTOS VEINTISEIS EUROS CON CINCO CÉNTIMOS).

ANEXO III. SEÑALIZACIÓN VERTICAL

1. Introducción

Debido a la variedad y complejidad de la señalización vertical la aplicación se ha centrado en la señalización vertical española, que viene legislada en el *REAL DECRETO 1428/2003, de 21 de noviembre, por el que se aprueba el Reglamento General de Circulación para la aplicación y desarrollo del texto articulado de la Ley sobre tráfico, circulación de vehículos a motor y seguridad vial, aprobado por el Real Decreto Legislativo 339/1990, de 2 de marzo*, más concretamente en el capítulo 4, subsección 4ª.

2. Señalización vertical

En referencia al Real Decreto mencionado en la introducción, de manera esquemática se pueden subdividir las siguientes señales verticales:

1. **Señales de advertencia de peligro**
2. **Señales de reglamentación**
   - *2.1. Señales de prioridad*
   - *2.2. Señales de prohibición de entrada*
   - *2.3. Señales de restricción de paso*
   - *2.4. Otras señales de prohibición o restricción*
   - *2.5. Señales de obligación*
   - *2.6. Señales de fin de prohibición o restricción*
3. **Señales de indicación**
   - *3.1. Señales de indicaciones generales*
   - *3.2. Señales de carriles*
   - *3.3. Señales de servicio*
   - *3.4. Señales de orientación*
   - *3.5. Paneles complementarios*
   - *3.6. Otras señales*

3. Codificación

Se puede observar, teniendo en cuenta el apartado anterior y el Real Decreto anteriormente mencionado, que cada señal vertical tiene su propia codificación, no obstante, se realiza una codificación propia con tal de hacer más fácil la consulta a la base de datos. Dicho esto pues, tenemos en consideración el apartado anterior, así tendremos el siguiente esquema:

Codificación			
Forma de la señal	**Tipo de señal**	**Subtipo de señal**	**Número de la señal**
1 dígito	1 dígito	1 dígito	4 dígitos

Tabla 34. Codificación de señales

- **Forma de la señal**
    1. Círculo
    2. Triángulo
    3. Cuadrado
- **Tipo de señal** (revisar apartado previo)
- **Subtipo de señal** (revisar apartado previo). En caso de no existir subtipo de señal, se implementará un "0" por defecto.
- **Número de la señal**. Viene dado por el número de referencia de la señal expuesto en el Real Decreto. Es de destacar que algunas señales contienen también una letra al final, por lo que se considerará cada letra como un número respecto a su posición en el abecedario. De no existir una letra final, se implementará como un "0" por defecto. De no existir más números a la izquierda, se completarán con ceros, en caso de que el número tenga menos de tres dígitos.

Nombre de la señal	Referencia	Codificación
Vía reservada para ciclos o vía ciclista	R-407a	1254071
Sentido obligatorio (recto)	R-400c	1254003
Sentido obligatorio (derecha)	R-400d	1254004
Intersección de sentido giratorio obligatorio	R-402	1254020
Obras	P-18	2100180
Niños	P-21	2100210
Ciclistas	P-22	2100220
Otros peligros	P-50	2100500
Calzada de sentido único (un carril)	S-11	3310110
Calzada de sentido único (dos carriles)	S-11a	3310111
Estacionamiento	S-17	3310170

Tabla 35.Codificación de señales utilizadas

ANEXO IV. EXPECIFICACIONES DEL DISPOSITIVO ANDROID

1. Introducción

En este Anexo se detallarán las especificaciones más relevantes con respecto a este TFG pertenecientes al dispositivo Android de la marca Sony de la gama Xperia y modelo Z1.

2. Especificaciones

Ilustración 50. Imagen de Sony Xperia Z1

- Sistema Operativo Google Android 4.4.2 (KitKat).
- Procesador de cuatro núcleos Qualcomm MSM8974 de 2,2 GHz.
- Cámara de 20,7 megapíxeles con sensor de imagen Sony Exmor RS® para móviles con flash LED por pulsos.
- Zoom digital de 8 aumentos.
- Pantalla TRILUMINOS™ Full HD de 5 pulgadas con motor de imagen X-Reality™ para móviles con 16.777.216 colores y 1920 x 1080 píxeles de resolución.
- Memoria RAM de 2GB.
- Memoria flash de hasta 16GB.
- Ranura de ampliación de tarjeta microSD™ de hasta 64 GB.

ANEXO V. MANUAL DE USUARIO

1. Introducción

Este manual de usuario es una guía rápida para comenzar a utilizar la aplicación **DSTr** en cualquier dispositivo Android superior a la versión 4.0.

Destacar que las opciones que permite la aplicación se dividen en dos grandes bloques, uno de detección y reconocimiento de señales y otro de gestión de base de datos.

2. Pantalla de inicio

Para acceder a la aplicación hay que presionar sobre el botón con el logo en el que debajo reza “DetectorSeñalesTráfico”. Posteriormente nos aparecerá una página de inicio temporal durante 1 segundo.

Ilustración 51. Pantalla de inicio de la aplicación

Posteriormente se mostrará una pantalla con el fondo en blanco, con una barra en la parte superior en la que aparecerá el nombre de la aplicación en la parte superior izquierda y un botón para seleccionar en la parte superior derecha.

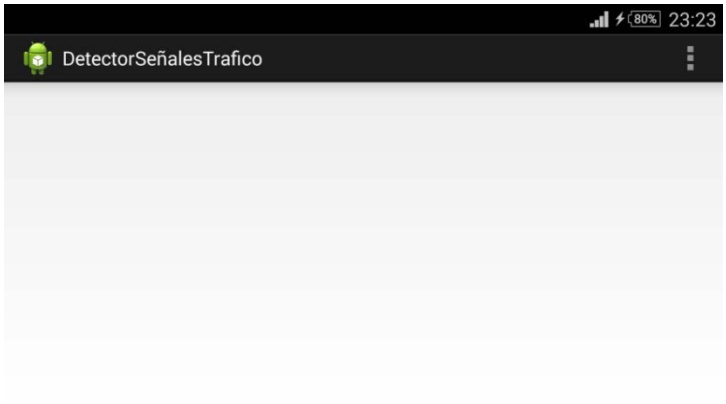


Ilustración 52. Pantalla principal de la aplicación

3. Pantalla de selección

Teniendo en cuenta la Ilustración 52, desplegamos el menú de selección, en el cual podemos ver las diferentes funciones de la aplicación.

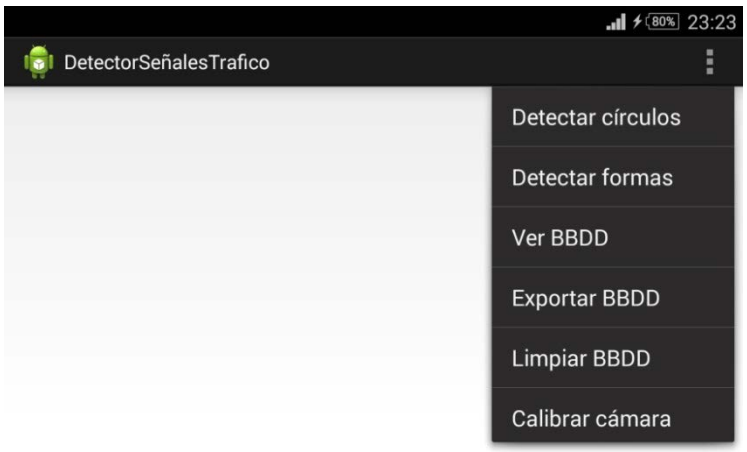


Ilustración 53. Pantalla principal con menú de selección desplegado

En ella se pueden ver diferentes opciones, que están englobadas en dos apartados, dentro de los que se encuentran estos subapartados. A saber:

- **Detección y reconocimiento de señales**
  - Detectar círculos
  - Detectar formas
  - Calibrar cámara
- **Gestión de base de datos**
  - Ver BBDD
  - Exportar BBDD
  - Limpiar BBDD

4. Detección y reconocimiento de señales

La opción de “Calibrar cámara” nos permite visualizar previamente los objetos a los que estamos enfocando. Es de vital importancia para comprobar que la cámara funciona de manera correcta con la aplicación, de lo contrario no llegará a detectar ningún tipo de señal.

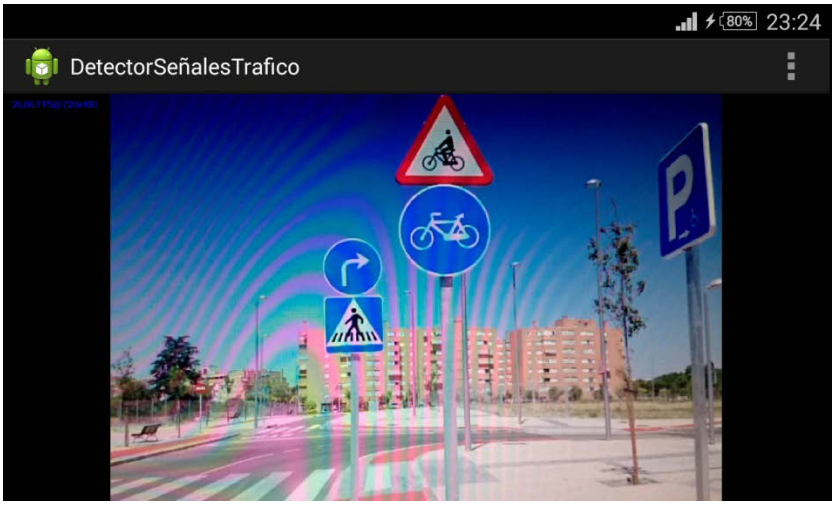


Ilustración 54. Opción de "Calibrar cámara"

La opción “Detectar círculos” nos permite detectar y reconocer algunos tipos de señales de forma circular. Para ello, tal como se muestra en la Ilustración 53 debemos elegir esta opción. Un ejemplo de esto se puede ver en la siguiente imagen.

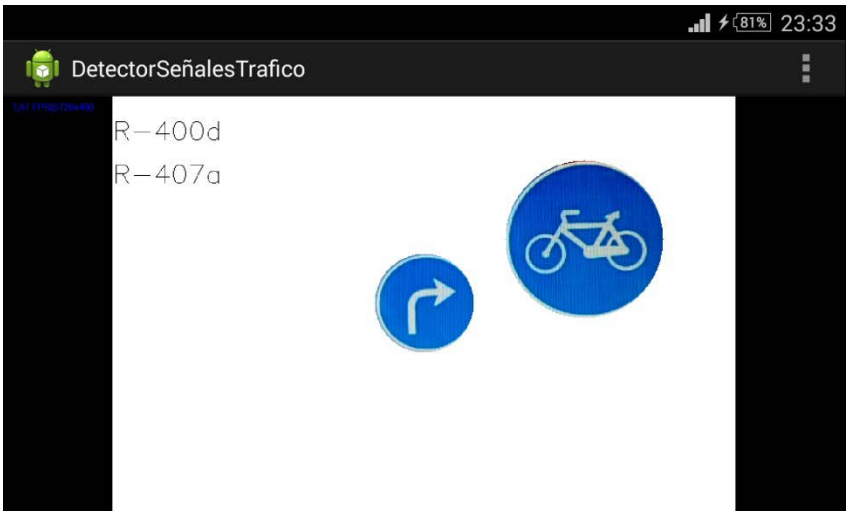


Ilustración 55. Opción de "Detectar círculos"

Aquí se puede ver que se han identificado dos señales con forma circular, concretamente la R-407a y la R-400d (ver Anexo III), y se muestran sus referencias en la parte izquierda de la pantalla.

La opción “Detectar formas” nos permite detectar y reconocer algunos tipos de señales de forma triangular o cuadrada. Para ello, tal como se muestra en la **¡Error! No se encuentra el origen de la referencia.** debemos elegir esta opción. Así pues, obtendremos lo que aparece en la siguiente imagen una vez que ésta sea detectada y reconocida.

Ilustración 56. Opción de "Detectar formas"

5. Gestión de bases de datos

Si se tienen habilitadas en el dispositivo Android la opción de GPS y/o Internet al reconocer algún tipo de señal, ésta es almacenada en una base de datos de la aplicación de manera automática con su identificador de señal (mirar Anexo III) y su geoposicionamiento (latitud y longitud).

Para ello podemos obtener la lista de registros guardados de la aplicación pulsando sobre “Ver BBDD”, mostrado en el desplegable de la Ilustración 53. Así obtendremos la siguiente imagen.

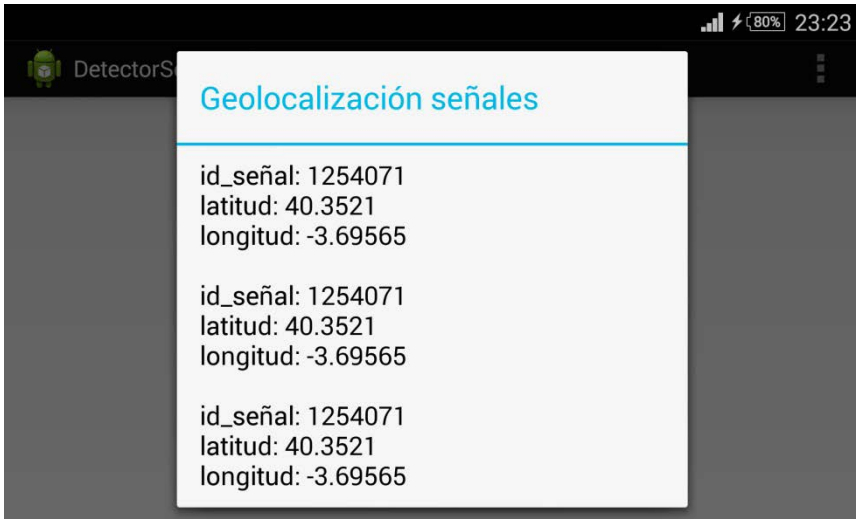


Ilustración 57. Opción de "Ver BBDD"

Así, los valores que contiene la Ilustración 57 son *id_señal*, que se corresponde con el identificador de señal tal como se expone en el Anexo III de esta memoria; *latitud*, que es la latitud obtenida por GPS, y la *longitud*, obtenida de la misma forma que la latitud. Nótese que puede haber datos duplicados, ya que la aplicación no se encarga de comprobar previamente si estos ya existen. El tratamiento de los datos se ha de producir de manera externa a la aplicación.

También se puede exportar la base de datos mediante la opción “Exportar BBDD” a una tarjeta externa de memoria para poder tratar sus datos.

Con la opción “Limpiar BBDD” se tiene la capacidad de limpiar los registros de la base de datos, dejando ésta a cero por si se requiriese.